IRE Transactions on ELECTRONIC COMPUTERS

Volume EC-6 DECEMBER, 1957 Number 4

Published Quarterly

TABLE OF CONTENTS

CONTRIBUTIONS

CORRESPONDENCE

PUBLISHED BY THE
PROFESSIONAL GROUP ON ELECTRONIC COMPUTERS

IRE Transactions® on Electronic Computers

Published by the Institute of Radio Engineers, Inc., for the Professional Group on Electronic Computers at 1 East 79th Street, New York 21, N.Y. Responsibility for the contents rests upon the authors and not upon the IRE, the Group, or its members. Price per copy: IRE-PGEC members, $1.20; IRE members, $1.80, nonmembers, $3.60. Yearly subscriptions rate: nonmembers, $17.00; colleges and public libraries, $12.75. Address requests to The Institute of Radio Engineers, 1 East 79th Street, N.Y. 21, N.Y.

Notice to Authors: Address all papers and editorial correspondence to J. P. Nash, Missile Systems Division, Lockheed Aircraft Corp., 3251 Hanover Street, Palo Alto, Calif. To avoid delay, 3 copies of papers and figures should be submitted, together with the originals of the figures which will be returned on request. All material will be returned if a paper is not accepted.

The Synthesis and Analysis of Digital Systems by Boolean Matrices*

JOSEPH O. CAMPEAU†

Summary—In this paper methods are described by which Boolean matrices can be used to synthesize digital systems. The matrices offer a means by which the design of such systems can be systematized much in the same way as do matrix methods when applied to electrical circuit design. They also present a means by which the problems of optimum logical design and programming can be approached.

Introduction

IN THE DESIGN of electrical circuits the use of matrix methods has been found helpful. The problems encountered in the design of electrical circuits are similar in some respects to those which exist in the design of digital systems.

It is, therefore, not unreasonable to expect that matrix methods might prove helpful when applied to digital systems.[1] This has been found true in two ways.

The first occurs in the design of digital systems where the process to be performed is already determined and where the only remaining problem is the writing of the logic to perform the task. In this type problem the matrix methods appear helpful. For example, one problem to be worked later in this paper concerns a system which is to add 3 or 5 to any number shifted into a bank of flip-flops, or multiply this number by 3 or 5, depending on the state of two-phase control flip-flops.

What the matrix methods do when applied to problems of the general type of which the above is an example is to make the work involved assume more of a "crank-turning" nature than would otherwise be the case.

The second way in which the matrix methods appear valuable is in an approach to the design of optimum digital systems and also in optimum programming.

Often, in the design of digital systems, more than one method can be determined which will accomplish the desired task. One of the methods, however, will require less diodes or flip-flops or both than do the other methods. It is the task of the person designing a system to devise methods to accomplish the desired task which will require the least amount of equipment. He tries to design an "optimum" system.

Much in the same way, once a general purpose digital computer has been built there remains the job of programming it to perform some specific process. By properly arranging the information in the memory, the programmer can often perform the program with less orders and in less time than he could have with some other arrangement.

By using matrix methods it is possible to reduce the problems of optimum design and programming to a system of simultaneous Boolean equations where the unknowns are the elements of the matrix which represents the desired optimum design or program.

In general, it has been found that these equations are too voluminous to be solved by the methods at hand, so that the results obtained are not of practical significance at the present time.

If, in the future, techniques are worked out which allow the solution of these equations, then a technique which will allow an analytical approach to optimum design and programming will be available.

In order to conserve space no detailed developments will be presented in this paper. A more complete treatment of the material can be found in the paper referenced above.

Boolean Matrices

Consider the following set of Boolean equations:

$$\begin{aligned} x_1 \quad + x_2 \quad &= y_1; \\ x_1\bar{x}_2 + \bar{x}_1 x_2 &= y_2. \end{aligned} \tag{1}$$

In the above set of equations the union or logical "or" and intersections, or logical "and" operations, are referred to rather than sums and products (as will generally be the case unless it is stated that arithmetical operations are intended). Eq. (1) can be written (expanded) into minimal polynomials:

$$\begin{aligned} 0\bar{x}_1\bar{x}_2 + 1x_1\bar{x}_2 + 1\bar{x}_1 x_2 + 1x_1x_2 &= y_1; \\ 0\bar{x}_1\bar{x}_2 + 1x_1\bar{x}_2 + 1\bar{x}_1 x_2 + 0x_1x_2 &= y_2. \end{aligned} \tag{2}$$

Notice that here all minimal polynomials appear multiplied by either a 1 or a 0.

Finally the coefficients of the minimal polynomials could be written along with the variables in the following form:

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}. \tag{3}$$

The components in the above vectors can only have values of one or zero and so the vectors can be called "Boolean vectors." The array of ones and zeros is not

* Manuscript received by the PGEC, August 27, 1956; revised manuscript received, August 28, 1957. Work on this paper was carried out while the author was employed at the Hughes Aircraft Co., Culver City, Calif.

† Litton Industries, Beverly Hills, Calif.

[1] For the original presentation of this material see J. O. Campeau, "The synthesis and analysis of counters in digital systems by Boolean matrices," Master's thesis, University of California, Los Angeles; June, 1955.

a matrix in the ordinary sense of the word. It will be called a "Boolean matrix" and is the array of the coefficients of the ordered minimal polynomials in (2). In the form shown above the original set of Boolean equations has been written as a "matrix" equation. Now we determine a method which will allow the passing from the Boolean matrix (3) to the set of (2).

In order to do this, a formalization of the method which will generate the minimal polynomial form of any system of Boolean equations is required. More specifically, in order to determine the Boolean matrix, the coefficients of the minimal polynomials which make up the function are required.

To begin, a row matrix A with elements a_j is defined:

$$a_j = j - 1 \qquad = 1, 2, 3, \cdots, 2^n \tag{4}$$

where the problem under consideration involves n variables (the above example involved two variables). In the case for $n=2$ the matrix A assumes the form

$$A = (0\ 1\ 2\ 3). \tag{5}$$

Then associated with the ordinary row matrix A we define another (Boolean) matrix A with n rows and 2^n columns with elements a_{ij} such that

$$a_j = \sum_{i=1}^{n} 2^{i-1} a_{ij}, \qquad = 1, 2, \cdots 2^n - 1, 2^n. \tag{6}$$

Here arithmetical addition and multiplication are intended and the a_{ij}'s are defined implicitly. This formula amounts to decimal-to-binary conversion.

Using (5) and (6) for $n=2$ the Boolean matrix A assumes the form

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{7}$$

The A matrix defined in (4) and (6) can then be used to pass from a system of equations such as (1) to a Boolean matrix (3). A "multiplication" of a Boolean vector x by a Boolean matrix B can be defined in the following manner.

Start with (1) generalized to n dimensions:

$$f_i(x_1, x_2, \cdots, x_n) = y_i, \qquad i = 1, \cdots, n. \tag{8}$$

The elements of a matrix B are determined from (8) by

$$b_{ij} = f_i(a_{1j}, a_{2j}, \cdots, a_{nj}) \qquad \begin{matrix} i = 1, \cdots, n \\ j = 1, \cdots, 2^n. \end{matrix} \tag{9}$$

Using (9), for example, the element b_{12} derived from (1) would be

$$b_{12} = f_1(a_{12}, a_{22}) = f_1(1, 0) = 1 + 0 = 1.$$

It can be seen that (9) provides a means by which the array of coefficients in (3) can be obtained from the set of (1). In fact, (9) is merely the desired formalization of the usual process used to obtain the minimal polynomials.

Then given the matrix equation

$$Bx = y, \tag{10}$$

a set of equations consisting of the unions of selected minimal polynomials of the form of (2) can be developed by using the relationship

$$y_i \overset{\Delta}{=} \sum_{j=1}^{2^n} b_{ij} \prod_{k=1}^{n} (a_{kj}x_k + \bar{a}_{kj}\bar{x}_k) \qquad i = 1, \cdots, n; \tag{11}$$

(the $\overset{\Delta}{=}$ stands for "equal by definition").

Eq. (11) will be used as the definition of the "multiplication of a vector x with elements x_i $(i=1, \cdots, n)$ by a matrix B with elements b_{ij} $(i=1, \cdots, n, j=1, \cdots, 2^n)$." In (11) the a_{kj}'s are the elements of the A matrix defined in (4) and (6). The $\prod$ term refers to intersections over the range of the dummy index k while the $\sum$ term refers to unions over the dummy index j.

Some comment is appropriate at this time concerning the above equation. The $\prod$ terms are each a particular minimal polynomial involving the variables $x_1, x_2, \cdots, x_n$. For example, consider the minimal polynomial generated for $j=2$ where the problem involves two variables x_1, and x_2 (so that $n=2$). The $\prod$ term in (11) for $j=2$ can be expanded to read:

$$\prod_{k=1}^{2} (a_{k2}x_k + \bar{a}_{k2}\bar{x}_k) = (a_{12}x_1 + \bar{a}_{12}\bar{x}_1)(a_{22}x_2 + \bar{a}_{22}\bar{x}_2).$$

From (7):

$$a_{12} = 1,$$
$$a_{22} = 0.$$

Thus the above expression becomes

$$\prod_{k=1}^{2} (a_{k2}x_k + \bar{a}_{k2}\bar{x}_k) = (1 \cdot x_1 + 0 \cdot \bar{x}_1)(0 \cdot x_2 + 1 \cdot \bar{x}_2)$$
$$= x_1\bar{x}_2.$$

This expression will be recognized as one of the minimal polynomials of the variables x_1 and x_2.

In (11) the minimal polynomials generated are each multiplied by the appropriate element of the Boolean matrix. It will be recalled that the elements of the Boolean matrix are defined to be the coefficients of the ordered minimal polynomials making up a set of Boolean equations. Thus by multiplying the $\prod$ terms (the minimal polynomials) by the correct coefficient and then summing the results (indicated by the $\sum$ over the j's) the original equations can be obtained.

In considering (10) and (11), the question might be asked, why use the $\prod$ terms (which involve the elements of the A matrix defined earlier) in order to get the required minimal polynomials? Furthermore, why define a special type of multiplication in order to expand the Boolean matrix of a system of equations? For example, given a set of variables $x_1, \cdots, x_n$, a vector could be defined whose components would be the ordered

minimal polynomials of the variables of the variables (this vector would have 2^n components). Then this vector could be combined with the Boolean matrix of the system using ordinary matrix multiplication and the desired system of equations obtained without using the special operation defined above.

The answer to the above question is that in the sections to follow, the Boolean matrix will be used to describe the operation of any digital system in which all operations are controlled by a clock (that is, any synchronous digital system). In using the Boolean matrix in this way it becomes necessary to consider repeated applications of the matrix, since each application of the matrix advances the digital system by one clock time.

For example, the following operation might be desired: $B(Bx)$. Here the B matrix is applied to the vector x. The result (whether the special multiplication is used or not) is a system of equations. There will be exactly n of these equations, one for each of the variables in the problem. The expression Bx is thus another vector with n components (just as was the original vector x).

It can be seen that in order for the second application of the B matrix to have any meaning at all a multiplication which deals with the elements of a vector x and generates the minimal polynomials as part of its operation is required.

Thus even though the first application of the B matrix can be accomplished using the usual definition of the multiplication of a vector and a matrix, this is not adequate when repeated applications of a Boolean matrix are to occur.

The multiplication defined in (11) transforms one Boolean vector with n components into another vector which also has n components so that repeated applications of any Boolean matrix can be accomplished.

As a side comment, it is also possible to define the multiplication of a vector x by a Boolean matrix B using modulo two addition and intersection:

$$y_i \overset{\Delta}{=} \sum_{j=1}^{2^n} b_{ij} \prod_{k=1}^{n} (x_{kj} \oplus \bar{a}_{kj}) \qquad i = 1, \cdots, n.$$

The above definition provides a slightly neater formula. Note that here the summation is taken to mean modulo two addition rather than union. It can be shown quite easily that for a given Boolean matrix the right side of the above equation and of (11) are identical.

Also it is interesting to note that the Boolean matrix for a given system will be identical to the "truth table" which can be developed for the same system.

The reason for this property will be discussed at greater length below.

Properties of Boolean Matrices

In the previous section the Boolean matrix was introduced along with the operation which combines any Boolean matrix B which is $n \times 2^n$ in size with a Boolean vector of n elements to produce a set of n Boolean equations. In this section the Boolean identity matrix, matrix addition, matrix multiplication, and the Boolean determinant and the Boolean inverse matrix will be presented.

All of these things were developed in an attempt to parallel as much as possible the type of operations encountered in ordinary matrix operations.

The Inner Product: In order to facilitate the proofs to follow it is convenient to introduce an abbreviation. Given two Boolean vectors α and β with components α_i and β_i, define

$$(\alpha, \beta) \overset{\Delta}{=} \prod_{i=1}^{n} (\alpha_i \beta_i + \bar{\alpha}_i \bar{\beta}_i). \tag{12}$$

This will be called the "inner product" of the two Boolean vectors.

From the definition of the identity matrix given in (4) and (6) it is obvious that if a_j and a_i are Boolean vectors whose components are respectively the jth and ith columns of the identity matrix then:

$$\begin{aligned} (a_j, a_i) &= 1 \qquad \text{for } j = i \\ &= 0 \qquad \text{for } j \neq i. \end{aligned} \tag{13}$$

Using the inner product (11) can be written in the form

$$y = \sum_{j=1}^{2^n} b_j(x, a_j) \tag{14}$$

where y and x are Boolean vectors whose components are respectively y_i and x_i $(i = 1, \cdots, n)$, and b_j and a_j are vectors whose components are respectively the jth columns of the B and identity matrices.

Matrix Addition: Define the union (addition) of two Boolean vectors x and y with elements x_i and y_i as a new vector with components

$$z_i = x_i + y_i \qquad i = 1, \cdots, n, \tag{15}$$

where

$$z = x + y.$$

Matrix addition is given as

$$d_{ij} = b_{ij} + c_{ij} \qquad \begin{matrix} i = 1, \cdots, n \\ j = 1, \cdots, 2^n \end{matrix} \tag{16}$$

where b_{ij}, c_{ij} and d_{ij} are the elements of the matrices B, and C, and D and

$$B + C = D.$$

From the above definition of matrix addition it can be shown that

$$(B + C)x = Bx + Cx,$$

but that $B(x+y) \neq Bx+By$ in general where B and C are any two matrices and x and y are any two vectors with the same number of components.

Matrix Multiplication: Eqs. (11) or (14) define what

is meant by the multiplication of a matrix C by a vector x to produce another vector y:

$$Cx = y.$$

Suppose now that it is desired to multiply the vector y by another matrix B:

$$By = B[Cx].$$

This operation can be performed on the vector $Cx = y$ using (14) again.

What is now required is a method of combining only elements of B and C in such a way as to produce a new matrix D with the property that

$$B(Cx) = Dx.$$

When this operation (whatever it is) has been developed it can be called "Boolean matrix multiplication" since it performs exactly the same function with Boolean matrices as does ordinary matrix multiplication with matrices. The above equation would then further be written:

$$B[Cx] = Dx = [BC]x.$$

From (4) and (6) it can be seen that no two columns of the A matrix are identical element for element. Consider any Boolean vector x. For each of the 2^n possible x's in n space

$$(x, a_j) = 1,$$

for one and only one j if the a_j's are the column vectors of the A matrix. This fact will now be used in the derivation of the formula for Boolean matrix multiplication.

Use (14) to express the matrix equation

$$B[Cx] = z$$

in vector form:

$$z = \sum_{j=1}^{2^n} b_j(Cx, a_j)$$

$$= \sum_{j=1}^{2^n} b_j\left[\sum_{r=1}^{2^n} c_r(x, a_r), a_j\right].$$

What will now be done is to show that all operations to be performed on the above equation are valid for any vector x in n space, and hence are valid in general.

It was mentioned above that only one column of the A matrix will match element for element with any particular vector x. Call this column a_p. In the above equation (x, a_r) is nonzero only for $r = p$. Hence all the terms for which $r = p$ are equal to zero and can be dropped out without affecting the validity of the equation:

$$z = \sum_{j=1}^{2^n} b_j(c_p, a_j).$$

Now since

$$(a_r, x) = \begin{cases} 1 \text{ if } r = p \\ 0 \text{ if } r \neq p \end{cases},$$

the right side of the above equation can be multiplied by (a_p, x) and the terms

$$\sum_{\substack{r=1 \\ r \neq p}}^{2^n}\left[\sum_{j=1}^{2^n} b_j(a_j, c_r)\right](a_r, x)$$

added without affecting its value.

$$y = \left[\sum_{j=1}^{2^n} b_j(a_j, c_p)\right](a_p, x) + \sum_{\substack{r=1 \\ r \neq p}}^{2^n}\left[\sum_{j=1}^{2^n} b_j(a_j, c_r)\right](a_r, x)$$

$$= \sum_{r=1}^{2^n}\left[\sum_{j=1}^{2^n} b_j(c_r, a_j)\right](a_r, x).$$

Now define a matrix D with column vectors d_r such that

$$d_r = \sum_{j=1}^{2^n} b_j(c_r, a_j) \qquad r = 1 \cdots 2^n. \tag{17}$$

Then the above equation becomes

$$y = \sum_{r=1}^{2^n} d_r(a_r, x)$$

and this is equivalent to the matrix equation

$$Dx = y,$$

where

$$B(Cx) = (BC)x = Dx = y.$$

Eq. (17) is the formula for Boolean matrix multiplication of any two matrices B and C to produce the product D. In terms of the elements of the matrices:

$$d_{ir} = \sum_{j=1}^{2^n} b_{ij} \prod_{k=1}^{n} (c_{kr}a_{kj} + \bar{c}_{kr}\bar{a}_{kj}) \begin{cases} i = 1, \cdots, n \\ r = 1, \cdots, 2^n. \end{cases}$$

In the above formula the a_{kj}'s are the elements of the A matrix defined in (4) and (6).

The Inverse Matrix and Other Properties: It can be shown that the A matrix defined in (4) and (6) is the identity matrix. Thus

$$Ax = x,$$

for any vector x. Furthermore, it is possible to define characteristic vectors of any Boolean matrix as being those vectors which for a given matrix have the property that

$$Be = e$$

where e is a characteristic vector of the matrix B.

The operations of multiplication of a vector or a matrix by a scalar are completely analogous to the usual operations and the formulas are the same.

It can be shown that the inverse of a given matrix B with column vectors b_j has an inverse B^{-1} (if it exists at all) with column vectors b_r^{-1} given by the following formula:

$$b_r^{-1} = \frac{1}{|B|}\sum_{j=1}^{2^n} a_j(b_j, a_r) \qquad r = 1, \cdots, 2^n$$

where a_j and a_r are column vectors of the identity matrix defined in (4) and (6).

The scalar expression "$1/|B|$" which multiplies the vector sum is developed so that it resembles the corresponding expression with ordinary matrices.

First define "division" of two Boolean variables a/b as being the problem: given $a=bc$ and knowing a and b, what is the value of c? Then define the Boolean determinant of a Boolean matrix B with column vectors b_j as

$$|B| = \prod_{i=1}^{2^n} \sum_{j=1}^{2^n} (b_j, a_i)$$

where the a_i's are the column vectors of the identity matrix.

It can be shown that if a Boolean matrix B has an inverse that $|B|=1$; otherwise $|B|=0$. Thus the expression "$1/|B|$" in the above formula for the inverse will have value $1/1=1$ if an inverse exists and value $1/0=$undefined if no inverse exists.

The Row Boolean Matrix

Given any Boolean matrix it is possible to define a row matrix. The row matrix is obtained by regarding each column of the Boolean matrix as a binary number. and by writing down the decimal equivalent to this number. A matrix B with elements b_{ij} becomes a row matrix B with elements b_j where

$$b_j = \sum_{i=1}^{n} 2^{i-1} b_{ij} \qquad j = 1, \cdots, 2^n. \tag{18}$$

An example will illustrate this. Consider the Boolean matrix presented in (3). This matrix and its row equivalent matrix have been shown below:

$$B = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$B = [\ 0 \quad 3 \quad 3 \quad 1\ ]. \tag{19}$$

In addition to this, given a Boolean vector x it is possible to reduce it to a single number by the same technique. For example, the vector

$$x = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ becomes } x = (1)$$

while the vector

$$x = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ becomes } x = (3).$$

In the above equations arithmetical multiplication and addition are to be used.

The Application of Boolean Matrices to the Design of Digital Systems

The discussion in this section will be limited to digital systems which operate synchronously; that is, systems in which all operations are initiated by a clock pulse which appears at a fixed rate. Furthermore, all the storage in the system will be assumed to be accomplished by identical delay elements. This does not sacrifice any generality since it can be shown that any synchronous digital system can be reduced to an equivalent system which will have only identical delay elements and logic. The total number of binary storage elements in the two systems will be the same, and the states assumed by the storage elements in the two systems will be identical at all times.

The Representation of Digital Systems by Boolean Matrices

The synchronous digital system composed of delay elements and logic can be described by a system of Boolean equations in the following way. The delay elements in the system can be called $Q_1, Q_2, \cdots, Q_n$, and the variable x_1, for example, can represent the state assumed by the delay element Q_1 during this interval. In general x_i $(i=1, \cdots, n)$ can be the present state of Q_i.

When the clock pulse appears, each of the n delay elements in the system will sample its input logic, and when the next clock pulse appears each delay element will have assumed the state, either "1" or "0" which its input logic had at the present clock pulse. The state the delay element Q_1 will have assumed when the next clock pulse appears can be called y_1, and in general y_i $(i=1, \cdots, n)$ can represent the next state of Q_i.

In this manner it is possible to represent the state of each delay element in the system at the next clock pulse in terms of the states of all the elements in the system at the present clock pulse.

Consider the following digital system shown in Fig. 1.

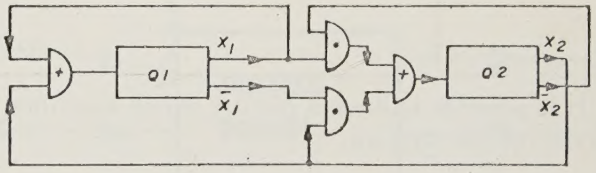


Fig. 1—Digital system.

The boxes labeled Q_1 and Q_2 are delay elements both with identical delays of 1 bit time. These delay elements sample the input logic on the left when the clock pulse arrives. When the next clock pulse arrives the output state of the delay element will be either a "1" or a "0" depending on what the input state was.

In each box Q_i, the output on the upper right is x_i $(i=1, 2)$ while the one on the lower right is $\bar{x}_i$. The element with the "+" performs the logical "and" or intersection operation.

The system of equations which give the state of the delay elements Q_1 and Q_2 at the next clock pulse (y_1 and y_2) in terms of their present states (x_1 and x_2) is then

$$x_1 + x_2 = y_1$$

$$x_1\bar{x}_2 + \bar{x}_1 x_2 = y_2.$$

It will be noted that this system of equations is identical to (1). It will be recalled that the Boolean matrix for this sytem of equations was

$$B = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

and that the equivalent row Boolean matrix is given from (18) as

$$B = [0 \quad 3 \quad 3 \quad 1].$$

This row matrix then corresponds to the digital system whose block diagram appeared in Fig. 1. It is instructive to trace through the states the system passes into when it is placed in various configurations. In order to do this write down the row identity matrix A. Directly beneath this write the row matrix for the digital system B:

$$A = [0 \quad 1 \quad 2 \quad 3],$$
$$B = [0 \quad 3 \quad 3 \quad 1].$$

Then to find what the next state of the system will be locate the present state in the matrix A. Directly beneath this in B will be found the next state of the system.

For example, directly beneath "3" in the A matrix is "1" in the B matrix. From (18), the "3" stands for $x_1=1$ and $x_2=1$. The "1" stands for $y_1=1$ and $y_2=0$. This result can be checked by referring to the block diagram of the digital system.

Table I summarizes the operation of the system.

TABLE I

Present State	Next State
0	0
1	3
2	3
3	1

It is possible to draw a picture which describes the behavior of the system:

Thus, if the system is placed in the state "0" (in which $Q_1=0$ and $Q_2=0$) it will remain in that state. If the system is placed in the state 2 (in which $Q_1=0$ and $Q_2=1$) it will move to state 3 ($Q_1=1$ and $Q_2=1$), and so forth.

The above example illustrates the use of the row Boolean matrix to determine the state diagram of any given digital system. The reason that the row identity matrix can be used with the row matrix form of the Boolean matrix of the system to determine the state diagram of the system is as follows. What is actually desired in determining the state diagram is a truth table of the system. The truth table for a system consists of all possible states the system can assume listed in a column and the corresponding transformed states listed opposite in a second column. From the definition of the row Boolean matrix given in (4) it is obvious that all possible states of the system are present in this special matrix.

Consider (14). This equation gives the matrix equation $Bx=y$ in terms of the vectors x and y and the column vectors of the B matrix and the identity matrix. It can be seen that for a given x the expression (x, a_j) is nonzero for one and only one value of j, say $j=p$. Thus essentially (14) for a particular x reduces to

$$y = b_p$$

where (x, a_j) is nonzero only for $j=p$. When the row identity matrix is written above the column B matrix it can be seen that this is equivalent to mechanizing (14). Thus for a given x we search through the identity matrix until a column is determined which matches x exactly (equivalent to determining the number p described above). The pth column of the matrix B then becomes the new state of the system, y.

The Synthesis of Digital Systems

In this section methods which allow the use of Boolean matrices in the synthesis of digital systems with prescribed operations will be introduced. The Boolean matrices offer a method of synthesis which deals only with the coefficients of the minimal polynomials making up the logic rather than with the minimal polynomials themselves. This eliminates the need for writing the minimal polynomials until the final logic has been determined.

The first problem to be considered will be that of designing a system which will replace any two numbers shifted into it with their product. In the simplest case, which is not trivial, the system desired would be one which accepts any two binary numbers between zero and three. The storage elements comprising the system can be called Q_1, Q_2, Q_3, and Q_4. Then the least significant bit of the first number will be stored in Q_1, the most significant bit of the first number in Q_2, the least significant bit of the second number in Q_3, and the most significant bit of the second number in Q_4.

In this case, since the storage elements are by the nature of the problem grouped two by two rather than all together, it is more expedient to write the identity row matrix A in the form shown below rather than in the usual manner. The least significant bit of the product is to appear in Q_1 and the most significant bit to appear in Q_4. Directly beneath each row in A the number it is to transform into is written down. The second row of numbers can be shown to be the B matrix for the system. Then the A and B matrices become

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \end{bmatrix},$$
$$B = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 0 \quad 2 \quad 4 \quad 6 \quad 0 \quad 3 \quad 6 \quad 9].$$

Notice that directly below the two elements which are present in the modified A matrix is their product. For example, in the last column the elements in the A matrix are 3 and 3. Directly beneath them in the B matrix is 9, the product 3×3. Then using (18) the B matrix corresponding to the row matrix B can be determined. This amounts to writing beneath each element in B a column in B in binary form. This matrix B is given as

$$B = \begin{bmatrix} 0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&1&1&0&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix}.$$

Notice, for example, that directly beneath the 9 in the B row matrix appears the column $\begin{bmatrix}1\\0\\0\\1\end{bmatrix}$ in the matrix B.

Eq. (11) can then be used to obtain the system of equations which specify the inputs to the delay elements in the system in terms of the present states of these delay elements:

$$x_1\bar{x}_2x_3\bar{x}_4 + x_1x_2x_3\bar{x}_4 + x_1\bar{x}_2x_3x_4 + x_1x_2x_3x_4 = y_1,$$

$$\bar{x}_1x_2x_3\bar{x}_4 + x_1x_2x_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3x_4 + x_1x_2\bar{x}_3x_4 + x_1\bar{x}_2x_3x_4 + \bar{x}_1x_2x_3x_4 = y_2,$$

$$\bar{x}_1x_2\bar{x}_3x_4 + x_1x_2\bar{x}_3x_4 + \bar{x}_1x_2x_3x_4 = y_3,$$

$$x_1x_2x_3x_4 = y_4.$$

These equations can be simplified to read:

$$x_1x_3 = y_1,$$

$$\bar{x}_2x_1x_4 + x_2x_3\bar{x}_1 + x_2x_3\bar{x}_4 + x_2x_1\bar{x}_3x_4 = y_2,$$

$$\bar{x}_3x_2x_4 + x_3\bar{x}_1x_2x_4 = y_3,$$

$$x_1x_2x_3x_4 = y_4.$$

These equations give the next state of each of the delay elements in the system in terms of the present states of these elements. The present state of the ith delay element Q_i is x_i and the next state of this delay element is y_i $(i=1, \cdots, n)$.

The logic for an equivalent system using what are called "JK flip-flops" can be developed using the formula:

$$J_i\bar{x}_i + \bar{K}_ix_i = y_i \quad i = 1, \cdots, n. \tag{20}$$

The logic using the JK flip-flops is then

$$Q_1\colon\ JQ_1 = 0$$

$$KQ_1 = \bar{x}_3$$

$$Q_2\colon\ JQ_2 = x_1x_4$$

$$KQ_2 = \bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_3 + x_1x_3x_4$$

$$Q_3\colon\ JQ_3 = x_2x_4$$

$$KQ_3 = x_1 + \bar{x}_2 + \bar{x}_4$$

$$Q_4\colon\ JQ_4 = 0$$

$$KQ_4 = \bar{x}_1 + \bar{x}_2 + \bar{x}_3.$$

The JK flip-flop is essentially a conventional flip-flop with inputs to each of the tube grids.

As a second example consider the following problem: it is desired to design a system with only four delay elements Q_1, Q_2, Q_3, and Q_4. Any number $X=0$, 1, 2, or 3 can be shifted in binary form into the elements Q_1 and Q_2. At the same time the delay elements Q_3 and Q_4 are also set into one of four states.

Then, depending on the states Q_3 and Q_4 were just set to, only one of the following things is to occur:

1) If $Q_3=0$, $Q_4=0$, replace X by $3X$.
2) If $Q_3=1$, $Q_4=0$, replace X by $5X$.
3) If $Q_3=0$, $Q_4=1$, replace X by $3+X$.
4) If $Q_3=1$, $Q_4=1$, replace X by $5+X$.

This problem may be summarized in a block diagram:

The steps in the problem solution are as follows:

1) Write down the modified identity row matrix A.
2) Develop the row matrix B by writing beneath each column of the modified identity matrix the desired result.
3) Develop the matrix B from B by using (18). This amounts to writing down beneath each of the elements of the B matrix its binary equivalent.
4) From (11) write the logical equations which describe the desired system. Alternately, since the elements of the B matrix are the coefficients of the ordered minimal polynominals making up the desired functions the equations can be written directly.

These steps have been carried out below:

$$A = \begin{bmatrix} 0&1&2&3&0&1&2&3&0&1&2&3&0&1&2&3 \\ 0&0&0&0&1&1&1&1&2&2&2&2&3&3&3&3 \end{bmatrix}.$$

$$B = [0\ 3\ 6\ 9\ 0\ 5\ 10\ 15\ 3\ 4\ 5\ 6\ 5\ 6\ 7\ 8\ ].$$

$$B = \begin{bmatrix} 0&1&0&1&0&1&0&1&1&0&1&0&1&0&1&0 \\ 0&1&1&0&0&0&1&1&1&0&0&1&0&1&1&0 \\ 0&0&1&0&0&1&0&1&0&1&1&1&1&1&1&0 \\ 0&0&0&1&0&0&1&1&0&0&0&0&0&0&0&1 \end{bmatrix}.$$

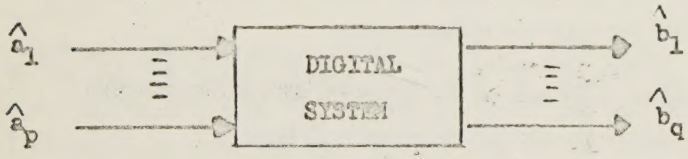
$$X_1\overline{X}_4 + \overline{X}_1X_4 = Y_1,$$

$$X_1\overline{X}_2\overline{X}_3\overline{X}_4 + \overline{X}_1X_2\overline{X}_3\overline{X}_4 + \overline{X}_1X_2X_3\overline{X}_4 + X_1X_2X_3\overline{X}_4 + \overline{X}_1\overline{X}_2\overline{X}_3X_4 + X_1X_2\overline{X}_3X_4 \quad X_1\overline{X}_2X_3X_4 \quad \overline{X}_1X_2X_3X_4 = Y_2,$$

$$X_3X_4(\overline{X}_1 + \overline{X}_2) + \overline{X}_3X_4(X_1 + X_2) + X_1X_3\overline{X}_4 + \overline{X}_1X_2\overline{X}_3 = Y_3,$$

$$\overline{X}_1X_2\overline{X}_4 + X_1X_2X_3 = Y_4.$$

It is possible to formalize the synthesis procedure which has been carried out above. Say the inputs to the system are the set of numbers $\hat{a}_1, \cdots, \hat{a}_p$ and that the outputs are a second set of numbers $\hat{b}_1, \cdots, \hat{b}_q$. Say



that each of the $\hat{a}_i$'s takes on a total of q_i values the $\hat{a}_i$'s. For each $\hat{a}_i$ determine a number k_i such that

$$2^{k_i-1} \leqq (\hat{a}_{ij})\max, < 2^{k_i}. \tag{21}$$

Even though a particular input $\hat{a}_i$ may not take on all values 0 to 2^{k_i-1} all values in this range will be considered in the discussion to follow.

Furthermore, it will be assumed that whatever the $\hat{a}_i$'s and $\hat{b}_i$'s represent, some code will have been used already to make them all positive integers. This does not sacrifice generality since it is common practice to employ such codes in all digital systems in the sense that all quantities are always represented by collections of i's and o's internally.

Since each of the inputs will be allowed to take on a total of 2^{k_i} $(i = 1, \cdots, p)$ values, and since there are a total of p inputs, the $\hat{a}_i$'s, there will be a total of $2^{\left[\sum_{i=1}^{p} k_i\right]}$ possible combinations of the $\hat{a}_i$'s, although some of the combinations will not be allowed (in the sense that each $\hat{a}_i$ may not take on all values from 0 to 2^{k_i-1}) in the actual system.

From an identity matrix with $\sum_{i=1}^{p} k_i$ rows as defined in (4) and (6) develop a second matrix A with elements a_{ij} where

$$a_{ij} = \sum_{n=p_i+1}^{p_i+k_i} a_{nj}2^{n-1-p_i} \qquad \begin{matrix} i = 1, \cdots, \sum_{i=1}^{p} k_i \\ = 1, \cdots, \sum_{i=1}^{p} k_i \end{matrix} \tag{22}$$

and the a_{nj}'s are the elements of the identity matrix.

$$p_i = \sum_{m=1}^{i-1} k_m \qquad \text{for } i \neq 1$$

$$= 0 \qquad \text{for } i = 1. \tag{23}$$

Now consider the matrix A. Each column of A will represent some combination of the inputs to the system, the $\hat{a}_i$'s. Some of the combinations represented by the columns of A will never occur since they may contain values of $\hat{a}_i$'s which will not occur. We will call these columns the "unallowed" columns in the discussion to follow.

Now form a second matrix B directly beneath A as follows. Each column of A represents some combination of the input variables, the $\hat{a}_i$'s. Beneath each column of A form a column of B where the element b_{ij} will be the appropriate value of the particular output variable $\hat{b}_i$. The order in which the $\hat{b}_i$'s are arranged in a particular column is unimportant, but the same ordering must be used for all columns of B.

Directly beneath unallowed columns of A any elements at all may be placed in the matrix B. This is to say, these elements are arbitrary.

The matrix B will have $2^{\left[\sum_{i=1}^{p} k_i\right]}$ columns and q rows (since there were to be q output quantities). In each row of B there will be some largest b_{ij}. For each row determine an m_i such that

$$2^{m_i-1} \leqq (b_{ij})_{\max} < 2^{m_i}. \tag{24}$$

Now using m_i determined above, form a matrix B from the matrix B. The elements of B, b_{nj}, are determined from the elements of B as follows:

$$b_{ij} = \sum_{n=q_i}^{q_i+m_i} b_{nj}2^{n-1-q_i} \tag{25}$$

where

$$q_i = \sum_{r=1}^{i-1} m_r \qquad \text{for } i \neq 1$$

$$= 0 \qquad \text{for } i = 1$$

and where

$$b_{nj} = 1 \text{ or } 0 \text{ only.}$$

Then B is the Boolean matrix of the desired logic. Those columns of B which correspond to "unallowed" columns of the matrix A may have any value. These columns can be used to simplify the logic which is to be developed.

Using (11) form the logic for the desired digital system. As was mentioned above, the unallowed columns of B, and correspondingly, the minimal polynomials developed from them, can be multiplied by either a 1 or a 0, depending on which produces the simplest logic.

Note that the above discussion also shows how any synchronous digital system can be represented by a Boolean matrix.

Given a synchronous digital system (in which all operations are initiated by a single pulse which comes at some nominal repetition rate), the Boolean matrix for the system can be determined as follows:

1) In the same manner as was described above, use (21)–(23) to determine a matrix A for the system.
2) Beneath each element in the matrix A place the corresponding element in the B matrix. The element in B is determined by subjecting the system to the particular set of inputs represented by each element of A.
3) Using (25) develop the Boolean matrix for the system.

It should be mentioned that in doing the above operations *all* storage elements in the digital system (including cells on a drum, for example) must be considered as part of the Boolean vector which gives the state of the system at any time.

Cells on the surface of a drum can be considered to be part of shifting registers made up of unit delay elements.

The Problems of Optimum Design and Programming

It was mentioned in the introduction that Boolean matrices offer an approach to the problem of optimum design and programming. In that section the meaning of optimum design and programming was outlined. Essentially, what is meant is that to perform a given task it is always desired to minimize the time taken by the equipment, the number of storage elements (flip-flops), and the number of logical elements (diodes, for example) required. In the case of the optimum program it is desired to so organize the information in the program that the problem can be worked in the minimum amount of time.

The material to be presented in this section is an outgrowth of material presented earlier. A complete treatment of this subject will not be attempted at this time, but will be deferred until a later date. This discussion will serve as an introduction to the concepts involved.

Optimum Design

In the previous section on the design of digital systems using Boolean matrices it was shown that any synchronous digital system could be represented by a single Boolean matrix.

Suppose that a design has been worked out for a given digital system which will perform a given task. After converting all parts of the system to delay elements and logic by using formulas of which (20) is typical, the Boolean matrix for the system can be written down. The number of variables involved in the system of equations the matrix represents will be equal to the number of delay elements in the system. From this it can be seen that the Boolean matrix which describes the digital system is in general extremely large.

The matrix representation for the given digital system will then be

$$Bx = y \tag{26}$$

where each component of the vector x is the present state of one of the delay elements in the system. The corresponding component of the transformed vector is the next state the delay element will assume.

The Boolean matrix B will be an array of 1's and 0's with n rows and 2^n columns when there are a total of n delay elements in the system.

In the operation of the digital system each clock pulse which advances things one step then corresponds to one multiplication of the vector x by the Boolean matrix B. After two clock pulses have occurred the states of all the delay elements in the system are given by z where

$$B(Bx) = B^2x = z. \tag{27}$$

After p clock pulses the states of all the delay elements are given by some vector z where

$$B^px = z \tag{28}$$

and where the vector x represents the initial states of all the delay elements.

In other words the "power" to which the matrix B is raised is determined by the number of clock pulses which have elapsed. Suppose that for the system under consideration here a total of p pulses must elapse before the desired process is completed. For example, for addition, one word time, in general, must elapse before the addition is completed and this might consist of 21 clock pulses.

The transformation which the given digital system has performed after p clock pulses is then given by the matrix B^p. Suppose that it is desired to determine whether or not there exists a second digital system which will perform the transformation B^p taking a different period, say q clock pulses, require the same number of delay elements x as did the original digital system with matrix B, and require a minimum number of diodes.

Call the matrix of this as yet undetermined digital system C. Then in order that the system with matrix C perform the desired transformation B^p after q clock pulses it is necessary that

$$B^px = C^qx = z \tag{29}$$

for all values of the initial condition x. This then means that element for element the following matrix equality must hold:

$$B^p = C^q. \tag{30}$$

If the system under consideration has n delay elements then the matrix (30) amounts to a total of $n2^n$ Boolean equations involving the known elements of B and the unknown elements of C.

The requirement that a minimum number of diodes be used in C can be taken into the problem by a second set of equations which specify the allowable matrices (only those which describe systems with a minimum number of diodes).

The solution to (30) which expresses the unknown elements of C in terms of the known elements of B and the above equations will then specify the matrix C and will represent the desired design with the minimum number of logical elements (diodes).

In considering the problem of optimum design at least two parameters are present: the number of decision and storage elements available. Generally it is desired to maintain a certain ratio between the number of storage and decision elements available. For example, it may be desired to hold the ratio at 25 diodes per flip-flop.

An important question which can now be considered is: to do a given job what is the minimum number of memory cells which are required? In order to answer this we must first differentiate between operations in the computer which sample the inputs and operations which process the inputs.

Thus a digital computer might accept a set of inputs and then spend many clock pulses processing the inputs to obtain an output. The multiplications required in the problem are performed sequentially and not at the same time.

All memory elements which are used to hold and sequence the inputs can be eliminated if all processing of the inputs occurs at the same time. Such an operation would be similar to an analog computer in which all operations occur simultaneously. The only memory absolutely essential is that which is necessary to store information from one input sample to the next. Thus, if any of the quantities developed depends on the past history of any other quantity present in the computer, some memory is required.

The amount of memory required is also a function of the amount of information present. Thus, if a particular function were always exactly predictable to within the last two or three bits, then only these bits would have to be stored. The rest of the quantity could be approximated as some function of time.

In order to determine the least amount of memory which will handle a given problem the following procedure can be used. Say a given digital system has matrix B and that it takes p clock pulses to perform the desired transformation B^p. The matrix will in general be n by 2^n in size when the total memory capacity of the digital system is n bits. What we desire to determine is whether a second matrix C can be constructed which will satisfy (30). The matrix C will then represent a digital system with n storage elements which does the same thing as B in q steps. In addition to this also impose the restriction on the matrix C that only a certain class of systems will be allowed. These allowable systems will still have n storage elements, but m of the n storage elements will 1) not be part of the output storage elements (those elements in which the "answer" appears) or part of the input elements (where the original data is placed) and 2) not be involved in any of the other n-m "active" storage elements in the system.

The problem is that C must have as many total storage elements as B in order to form the matrix equality shown in (30). In order to determine whether a system with only m storage elements can also accomplish the transformation B^p, what is done is to specify that m of the storage elements in C will have nothing to do with the rest of the system.

Such storage elements, since they have no part in the system, can be removed from the system without affecting its operation. It is not difficult to specify the equations which will only be satisfied by the elements of matrices which satisfy the above requirements. For example, for $n=2$, all matrices C for systems in which one of the two storage elements has nothing to do with the second storage element will have the following form:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & \bar{c}_{21} & \bar{c}_{22} \end{bmatrix}.$$

The fact that

$$c_{23} = \bar{c}_{21},$$
$$c_{24} = \bar{c}_{22},$$

can be shown to be equivalent to the requirement that X_1 does not appear in the logic for X_2. This can be shown from (11).

By the use of the above technique it is possible to set up a set of simultaneous Boolean equations. The unknowns in the equation will be the elements of the matrix C which will satisfy (30) so that it performs the same function as does B and also has m "unused" elements which can be dropped out.

Various values of m can be tried in this process, starting from $m=n$, until an m is determined for which a solution can be found to the above mentioned system of equations. A system with n-m storage elements will then be the one which can accomplish the transformation B^p with the minimum number of storage elements. Generally, of course, storage is required in any system in which the values any variable will take on as a function of time are not completely predictable and where it is desired to generate a function which depends on the previous values of the variable. When the values are completely predictable, "fixed" memory which cannot

be altered can be used, as opposed to the use of a changeable memory. This type memory can be contained in the logic for the system.

Optimum Programming

The problem of optimum programming arises after a general purpose digital computer has been designed and built. The programmer is given some specific problem which the digital computer is to perform. He then has the job of organizing the information in the memory and of causing the operations to be sequenced so that the required problem can be solved in a minimum time. In the sense that the digital computer is already designed, the programmer's bounds are fixed. He must work within the order code provided with the computer. Suppose the programmer has already written a program which will accomplish the desired job. This program specifies the initial state of the digital computer.

Start with all the delay elements in the computer in the "0" state and write the program into the computer's memory. This "writing" of the program can be represented by the application of a matrix C to a vector all of whose components are zero (call this vector x). Then the vector which will represent the computer with the program written will be y where

$$Cx = y. \tag{31}$$

The matrix C might be called the "program" matrix since it writes the program into a computer.

As in the case of the optimum design, reduce the given digital computer to an equivalent system which will contain only delay elements. Each storage cell on the drum, if the computer uses a drum, will then be a single delay element. Then construct a single vector x whose components are the delay elements comprising the computer. The logic in the computer acts at each clock pulse to transform the vector x into another vector y whose components are the respective states of each delay element in the computer at the next clock pulse. The matrix which represents this logic can be called B.

At each clock pulse the logic in the computer transforms according to the matrix B introduced above. After one clock pulse has gone by, for example, the state of the computer can be represented by z where

$$BCx = z. \tag{32}$$

In other words, the matrix C describes the program written into a computer all of whose storage elements are initially in the "0" state, and the matrix B represents the action the computer has taken on the program after one clock pulse.

Suppose the program takes p clock pulses to perform its task. After p clock pulses the state of all the memory elements in the computer can be given by some vector z where

$$B^pCx = z. \tag{33}$$

It is desired in the design of the optimum program to determine whether or not there exists a second program which will accomplish the same task but which will required less time. Call the program matrix for this unknown program D, and call the desired program time q. Then if the computer starts with all its memory elements in the "0" state, the result of writing the program as in (31) will be a vector y where

$$Dx = y. \tag{34}$$

After q clock pulses have elapsed, the state of all the memory elements in the computer can be represented by a vector z, similar to (33) where

$$B^qDx = z. \tag{35}$$

Now in order that both programs produce identical results, it is necessary that the results z in (33) and (35) both be the same. In order that this be true it is necessary that element for element the two matrices given below be equal:

$$B^qD = B^pC. \tag{36}$$

As in the case of the optimum design of a digital system, the above matrix equation amounts to a system of simultaneous Boolean equations involving the known elements of B, C, and the known numbers p and q. The unknowns in this system of Boolean equations will be the elements of the as yet undetermined program matrix D.

The solution to the matrix equation, the program matrix D, will then represent the desired optimum program. If no solution to these equations can be found, then no program exists which will perform the desired task in q clock pulses.

Simulation of Transistor Switching Circuits on the IBM 704*

R. J. DOMENICO†

Summary—When the configuration of a circuit and the equivalent representations of the transistors are known, a computer program can be written to yield the performance of the circuit and the mean values of the circuit parameters. Nonlinearity of the transistors is accounted for by piece-wise linearization of an equivalent circuit. Rules of interconnection have been devised to combine this procedure efficiently with the manipulation of the matrix equations that define the linear external circuitry. The general method can be extended to combinations of basic circuits.

Introduction

IN transistor switching circuits, certain combinations of device characteristics are known to be most detrimental to proper circuit performance. These combinations are relatively hard to find in new transistors, and usually occur after the device has been in operation for thousands of hours. A method of simulation has been developed to design transistor switching circuits with an over-all reduction in design time. This method permits the designer to observe the operation of his circuit with a transistor that has end-of-life characteristics allowing him to obtain more accurate performance data. The transient behavior of a transistor and its associated circuitry may be analyzed for any type of input excitation and under any load conditions.

The Transistor Equivalent Circuit

The transistor may be classified in the general class of four-terminal networks as either a linear active, or nonlinear active network, depending on the amplitude of excitation applied. In general, input signals of greater than five to ten millivolts will put the transistor into the nonlinear active class. As the amplitude of the excitation is increased to values normally encountered in switching circuits, nonlinear circuit-analysis techniques must be used to accurately describe its operation. Of the many methods advanced to analyze nonlinear systems, it appears most feasible, for digital-computer applications, to linearize the transistor piece-wise, if a good small-signal representation of the transistor can be made available.

Many good small-signal equivalent circuits have appeared in the literature. Of these, a pi equivalent presented in "Transistor Electronics" appears to be most adaptable.[1]

The equivalence has been modified from that shown in the text, to conform with the requirements needed for a switching circuit. These modifications will be discussed later. The final form used, which has proven very successful, is shown in Fig. 1. The current i_h,

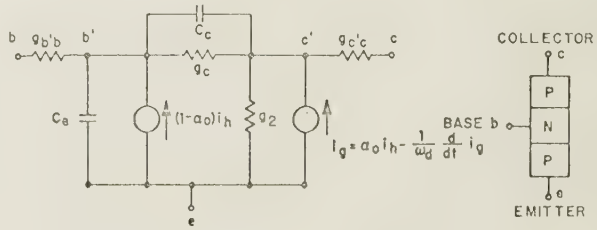


Fig. 1—Equivalent circuit for a *p-n-p* transistor.

which may be termed a minority carrier current, is related to the emitter-base voltage through the diode equation

$$i_h = i_0(e^{\Lambda V_{eb'}} - 1) \tag{1}$$

where

$$\Lambda = \frac{q}{KT} = \frac{1}{0.026 \text{ volts}}$$

at room temperature, and i_0 is the minority carrier current which flows when the diode is reverse biased. The generator which is connected between emitter and base is modified by $(1-\alpha_0)$, since the configuration is derived for common emitter. However, the circuit is equally valid for both common base and common collector. The current generator was included to account for the emitter-base voltage drop of 0.2 to 0.3 volts when the transistor is conducting. Accurate values for this voltage are necessary when predicting the response of a network driven from a voltage source.

If the diode equation is differentiated with respect to $V_{eb'}$, we obtain a conductance

$$G = \frac{d}{dV_{eb'}}, i_h = \Lambda i_0 e^{\Lambda V_{eb'}} = \Lambda(i_h + i_0). \tag{2}$$

This conductance is identical to the conductance discussed in "Transistor Electronics"[2] and defines the slope of the diode curve at any point.

The emitter capacitance C_e can be considered the parallel combination of two condensers. These are the barrier capacitance C_{TE} and the diffusion or storage capacitance C_s.

* Manuscript received by the PGEC, June 1, 1957; revised manuscript received, September 30, 1957. This paper was presented at the Second National Simulation Conf., Houston, Tex., April 12, 1957.

† IBM Corp., Poughkeepsie, N. Y.

[1] Lo, Endres, Zawels, Waldauer, and Cheng, "Transistor Electronics," Prentice-Hall, Inc., Englewood Cliffs, N. J., p. 264; 1955.

[2] *Ibid.*, p. 241.

$$C_e = C_{TE} + C_s. \tag{3}$$

The barrier capacitance C_{TE}[3,4] results from a contact potential formed by the junction of two opposite impurity-type semiconductors, P and N type for the emitter and base, respectively. This potential determines a depletion or transition region which acts as a condenser whose plates are the boundaries of the region. An expression which relates the capacitance to the applied voltage is

$$C_{TE} = \frac{K}{(V_0 - V_{eb})^n} \tag{4}$$

where V_{eb} is the applied emitter-base voltage; V_0 the contact potential; n a constant dependent on the type of junction; and K a function of emitter area, dielectric constant, conductivity, and width of the depletion region. For a drift transistor, typical values[5] of these constants are $V_0 = 0.5$ volts, $n = 0.4$, and $K = 15$ uuf-(volts)^n.

The storage capacitance acquires a charge equivalent to that stored in the base region when the transistor is conducting. An expression which relates this capacitance to i_h is

$$C_s = \frac{G}{2\pi f_s} = \frac{\Lambda(i_h + i_0)}{2\pi f_s} \tag{5}$$

where f_s is independent of i_h.[6]

The common-base cutoff frequency f_{ace} is related to two frequencies f_s and f_E, using the expression[6]

$$\frac{1}{f_{ace}} = \frac{1}{f_E} + \frac{1}{f_s} \tag{6}$$

where $1/f_s$ and $1/f_E$ are the time constants of the storage capacitance, the barrier capacitance, and the emitter conductance G.

$$f_s = \frac{G}{2\pi C_s} \tag{7}$$

$$f_E = \frac{G}{2\pi C_{TE}}. \tag{8}$$

Eq. (6) may be rewritten as

$$f_{ace} = \frac{f_s}{1 + \dfrac{f_s}{f_E}}. \tag{9}$$

Since f_s is independent of i_h, it can be shown, using (9), that f_{ace} approaches f_s as i_h is increased. For values of i_h greater than 5 to 6 ma, the following relationship holds:

$$f_{ace} \cong f_s. \tag{10}$$

This value of f_s is used in (5) to determine the storage capacitance.

While f_s is independent of i_h, it is dependent on the collector voltage. This dependence is introduced by the decreasing of the effective base width with increasing collector voltage.[6]

Using a derivation similar to that for C_{TE}, it can be shown that the collector capacitance is related to the collector voltage V_{cb} by

$$C_c = \frac{K}{(V_0 - V_{c'b'})^n} \tag{11}$$

where K, V_0, and n are as defined for C_{TE}.

The time constant $1/\omega_d$[7] adjusts the phase characteristic of the collector generator which would otherwise act as a minimum phase network. Measurement of the common base short-circuit current gain of the transistor shows that the phase angle is actually greater than 45° when the magnitude is down 3 db from the low frequency value. To a good approximation,

$$\omega_d = \frac{n'G}{C_e} \tag{12}$$

where n' is a constant for a transistor with a value between 2 and 3.

The conductances g_c and g_2 are of secondary importance in the transient solution. Both values are very small and experience has indicated that they can be approximated by constant values measured at a current and voltage close to the operating point.

The conductances $g_{b'b}$ and $g_{c'c}$ are bulk conductivities in the base and collector, respectively. The base bulk conductance is very important to the transient response while $g_{c'c}$ appears to have little or no effect when the transistor is kept out of saturation. The value of $g_{b'b}$ is obtained from a measurement of common-emitter input impedance with the collector short circuit. The measurement is made at 30 mc. Plots have been made as a function of collector current and voltages and the conductance has been found to be relatively constant.

The two remaining nonlinear parameters of the transistor are the current gain and the cutoff frequency. Both of these can be measured as a function of emitter current and plotted to determine a functional relationship for them.

The system of differential equations which describe the equivalent circuit may be written in nodal form in terms of the difference of voltages.

[3] L. J. Giacoletto and John O'Connell, "A variable capacitance germanium junction diode for uhf," *RCA Rev.*, vol. 17, pp. 221–238; June, 1956.

[4] R. D. Middlebrook, "An Introduction to Junction Transistor Theory," John Wiley & Sons, Inc., New York, N. Y., pp. 58–64, 160–163; 1957.

[5] D. DeWitt and A. Berger, IBM Product Development Lab., Poughkeepsie, N. Y. (private correspondence); August, 1957.

[6] A. L. Kestenbaum and N. H. Ditrick, "Design, construction, and high frequency performance of drift transistors," *RCA Rev.*, vol. 18, pp. 16–19; March, 1957.

[7] Lo, *et al.*, *op. cit.*, pp. 251–253.

$$\frac{d}{dt}(V_{c'} - V_{b'}) = \frac{1}{C_c}[ig + V_e g_2 + V_c g - V_{c'}(g_2 + g_{c'c}) - (V_{c'} - V_{b'})g_c] \quad (13)$$

$$\frac{d}{dt}(V_e - V_{b'}) = \frac{1}{C_e}\left[(V_{b'} - V_b)g_{b'b} - (1 - \alpha_0)i_h - (V_{c'} - V_{b'})g_c - C_c\frac{d}{dt}(V_{c'} - V_{b'})\right] \quad (14)$$

$$\frac{di_y}{dt} = \omega_d[\alpha_0 i_h - i_y]. \quad (15)$$

The value of i_b, i_c and i_e may be obtained by first integrating the differential equations for the differences of the internal node voltages. The resulting difference voltages and (16) through (20) are used to obtain i_b, i_c, and i_e.

$$V_b' = V_e - (V_e - V_{b'}) \quad (16)$$

$$V_{c'} = (V_c' - V_{b'}) + V_{b'} \quad (17)$$

$$i_b = (V_{b'} - V_b)g_{b'b} \quad (18)$$

$$i_c = (V_{c'} - V_c)g_{c'c} \quad (19)$$

$$i_e = i_b + i_c. \quad (20)$$

Two requirements must be satisfied before attempting a successful piece-wise linearization of a nonlinear system: the coefficients of the equation as a function of the operating point must be known; and sufficiently small intervals of integration must be maintained so that the linear assumption is not violated. The functional relationships for the coefficients can be determined from direct measurement and/or from the physical considerations discussed previously. A numerical method of integration is ideal for limiting the integration interval if the maximum value is directly controlled as input information.

The Network Analysis Program

A computer program which will meet this requirement is the SHARE program PK-NIDA (a numerically integrating differential analyzer). The mathematical basis for solution of the differential equations is Adam's method of numerical integration. Several logical features of the routine control the accuracy and speed of solution. The maximum integration interval and the allowable error are under control of the user. Integration may proceed at an interval which ranges between the assigned maximum interval and 2^{-16} of this value. The allowed error specifies the accuracy required between successive predicted and corrected values of the variables. For example, if an allowed error of 2^{-10} is used, accuracy to ten binary places is required. If predicted and corrected values disagree within these ten binary places, the program halves the integration interval, reverts back to the last good values, and proceeds with this smaller interval. If predicted and corrected values agree to 16 or more binary places, the program will double the integration interval and the integration continues at this larger interval. This halving or doubling of the interval can occur at any time during the integration process, except when first starting or when severe discontinuities are encountered. For these cases a set of starting equations are employed which use an interval 2^{-16} of the maximum. The values of the variables and their derivatives are stored in tables which contain the last eight computed values. These tables are continually updated as new values are determined, with the latest values placed at the top of the tables.

In the investigation of methods of evaluating actual circuit performance, it would appear on first sight that the method for analyzing the nonlinear transistor is incompatible with the analysis of the linear external circuitry. Manipulation of matrix equations is ideal for linear systems, while a method such as the one just described is best for a nonlinear system. However, an efficient combination of these methods has been achieved by a program which separates the transistors from the remainder of the circuit and produces a solution based on the most efficient analysis for each part.

The linear circuit elements are converted to matrix form in accordance with a set of simple rules which may be conveniently programmed.[8] Fig. 2 shows a simple RC

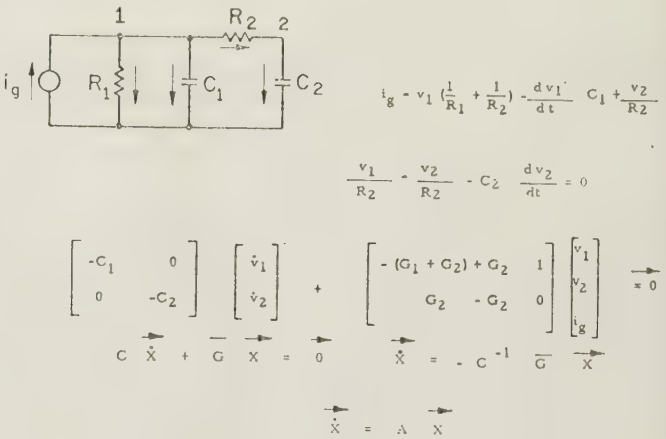


Fig. 2—A simple RC network and generator.

network and generator used to demonstrate this method. The differential equations for the system are shown. If the value of the generator is known, the network may be solved in matrix form by

$$C\dot{X} + \overline{G}\dot{X} = 0$$

where

C is the capacitance matrix shown;
$\overline{G}$ is a combination of a conductance matrix and a unit matrix;
$\dot{X}$ is a rate vector;
X is a current and voltage vector.

[8] J. C. Morgaun, Share Routine PE-CAML, IBM Product Dev. Lab., Poughkeepsie, N. Y.

If the C matrix is inverted and a matrix multiplication performed, the final expression $\dot{X}=AX$ is obtained, where A is the result of premultiplication of the $-\overline{G}$ matrix by the C^{-1} matrix. For a machine to perform these operations, it must be given a precise description of the circuit as input information. This description must include the following:

1) The number of independent nodes (so that the size of the C matrix can be determined).
2) The number of inductive currents.
3) All branch elements and the direction of positive current flow through them.

It is worth noting that the C matrix cannot be singular since it must be inverted. The singularity indicates that there is a dependent node associated with this matrix, and this dependency can be eliminated only if some stray capacitance is inserted at the node.

The forming of this matrix equation is performed entirely by the machine, based on the input data. Information is punched into cards, each card containing the type of element, its magnitude, the direction of positive current flow, and the nodes to which it is connected. With these data, the machine forms the C matrix, $-\overline{G}$ matrix, and the A matrix. Since the final A matrix contains only elements of the linear system, it is unnecessary to recompute it unless some component value is changed. As an example, we can consider how the machine forms the C matrix for this network. Since there are two independent nodes, this information is used to determine a 2 by 2 matrix. The condensers are located in the matrix by indicating the nodes to which they are connected, and enumerating these nodes in the order that determines positive current flow. For this network, the information is: C_1 between nodes 1 and 0, and C_2 between nodes 2 and 0. These condensers are then inserted with a negative sign in locations 1, 1 and 2, 2 of the matrix. If a condenser had been located between nodes 1 and 2, with positive current in this direction, it would have been inserted negatively in locations 1, 1 and 2, 2 and positively in locations 1, 2 and 2, 1.

If a transistor switching circuit like the one shown in Fig. 3 is considered, the circuit may be divided into two parts. One part contains the external linear circuitry, and the other the nonlinear transistor. Since it is known that matrix equations afford the most efficient method for handling linear systems, a look at the general matrix equation will reveal a method of linking the transistor to this solution to obtain a solution for the entire system.

Fig. 4 shows this equation along with the modified circuit which will be considered. The matrix equation is $\dot{X}=AX$, where the rate vector $\dot{X}$ contains all the derivatives of the independent node voltages and currents. The vector X contains the magnitude of these same currents and voltages, as well as the fixed bias voltages and the voltages or currents supplied by generators. The A matrix contains the values of all the passive elements of the linear circuit. Integration of the rate vector

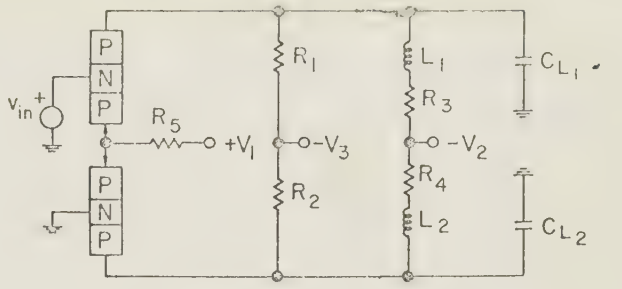


Fig. 3—A transistor switching circuit.

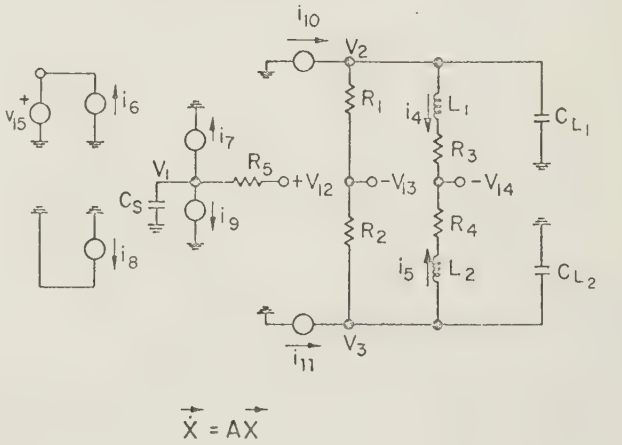


Fig. 4—Modified configuration of the switching circuit.

over a small interval of time results in new values for the X vector. If the integration interval is controlled so that the X vector, containing the transistor node voltages, does not change enough to violate the linear treatment of the transistor, it is possible to justify the insertion of a current generator at each node to which the transistor is connected. These current generators are shown in the circuit diagram. The X vector will now contain values for the node voltages and currents and the external voltage generators, plus the transistor current generators.

Fig. 5 shows the actual matrix which would be constructed by the machine for this circuit. It can be seen that the equations for the inductive currents have been added to the A matrix after the multiplication of the $-\overline{G}$ matrix by the C^{-1} matrix.

The interconnection between the solutions of the linear and nonlinear parts of the circuit is effected through the X vector. Initially, the transistor current generators are held constant and the rate vector is evaluated. This rate vector is then integrated over a controlled interval of time to obtain new values for the X vector. The voltages in the X vector which are associated with the nodes to which the transistors are connected are transferred to the routine which solves the transistor equations. Using these voltages, the differential equations are then evaluated and integrated over the same time interval to produce new values for the current generators. These new values are inserted back into the X vector, and a transfer is made to the matrix

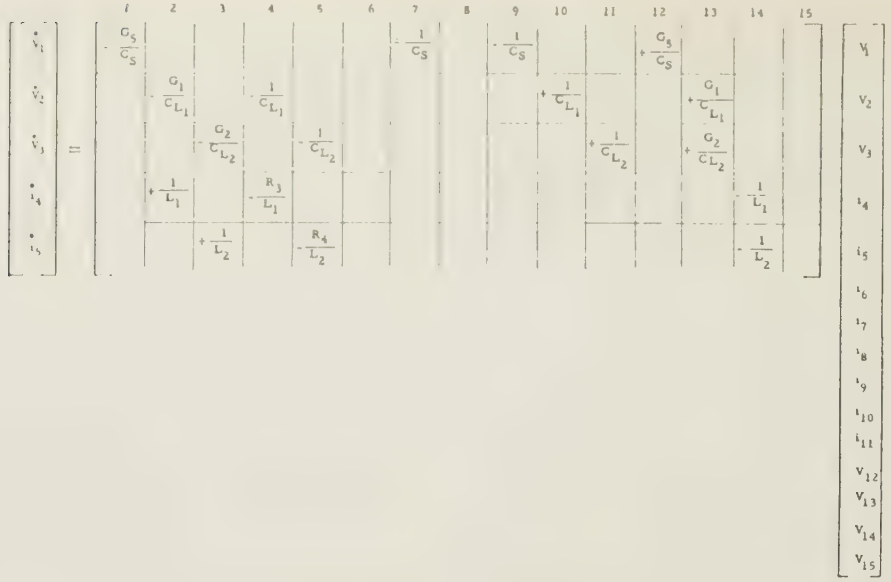


Fig. 5—Constructed matrix equation.

routine where a new rate vector is computed. This iterative process is continued until no further change is obtained at the output for some input excitation.

Since minimum storage requirements and programming ease are of primary importance in this simulation method, these two points have been considered in developing the program. The first has been accomplished by accommodating all transistors with one NIDA routine. The second has been achieved by locating automatically all variables pertinent to a transistor, given only the number of the transistor and the location of its node voltages and current generators in the X vector.

To initialize the system, the above information must be given for each transistor. In addition, the user must supply the following: data needed to create the A matrix, the maximum integrating interval, the allowed error, the number of transistors in the circuit, and parameter values for the transistors.

With the matrix equation written and the transistor subroutine initialized, the program transfers to the linear circuit routine and computes the values of the rate vector from the matrix equation. It then proceeds to predict and correct new values for the X vector over a fixed integration interval equal to the maximum value inserted into the program. These new values are then stored back into the X vector and the program transfers to the transistor.

This routine handles the transistors one at a time. The values of the base, emitter, and collector voltages are obtained from the X vector and are used first to modify any parameters which are a function of them. Other parameters are modified as a function of the emitter current.

Since only one NIDA routine is used for all transistors, it is important to retain the integration interval that was used on the transistor in the previous pass. This is necessary since the eight values stored in the table for each variable are spaced in time according to the interval. The importance of this becomes apparent if we consider the operation of two successive transistors in the routine. If the first of the two transistors were operating in a region where no discontinuities were present, NIDA would be using an integration interval near or equal to the maximum value. If the second transistor had in the meantime encountered some difficulty, it would be using much smaller integration intervals. Assuming that the routine had just completed the first transistor and passed on to the second, NIDA would still have a large interval stored. It would interpret the tables as being spaced in time according to this large interval instead of the small interval for which they were computed. Using the wrong interval would give an erroneous value for the next point. With the proper interval restored into NIDA, by address modification, the differential equations are evaluated and integrated, and the tables updated. The program will continue with this transistor until it has progressed exactly one maximum integration interval. It is important to note here that this interval must be maintained equal for all transistors in order to maintain a time relationship between the linear and nonlinear part. When the computation of the last transistor is completed, the values of all emitter, collector, and base currents are stored back into the X vector, and the program returns to the matrix equations.

Figs. 6 and 7 show the computed response of the circuit in Fig. 3. A voltage source with a fall time as indicated was used to drive the circuit in both cases. Identical transistors with the indicated characteristics were used to compare the effect of a change in capacitance on the collectors of both transistors. As would be expected the addition of the capacitance reduced the overshoot and increased the rise time, but did not affect the turn-on delay caused by having the transistor heavily cut off.

The program thus far has attempted to show the feas-

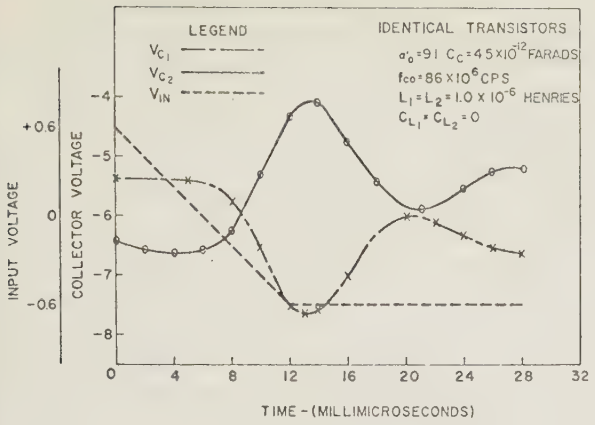


Fig. 6—Unloaded response of the circuit in Fig. 3.

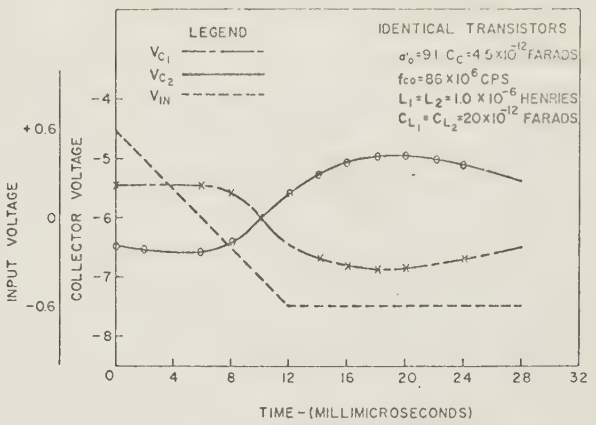


Fig. 7—Loaded response of the circuit in Fig. 3.

ibility of such an approach in determining the response of transistor switching circuits. It should not be inferred that the method is a substitute for the preliminary design of circuits, or a means of gaining insight into the operation of a circuit. It should be considered instead as a substitute for the final bench setup used to obtain data. The program needs further work before its use can be widely advocated. At present the program requires 15 to 20 minutes to obtain a complete response, and its capacity is a circuit containing up to ten transistors. The next phase of this program will concentrate on methods of reducing the computation time and making the routine more flexible.

Acknowledgment

The work reported here is being done at the IBM Poughkeepsie Product Development Laboratory. The author wishes to acknowledge the valuable assistance of the transistor computation group in completing this phase successfully.

An Optimum Character Recognition System Using Decision Functions*

C. K. CHOW†

Summary—The character recognition problem, usually resulting from characters being corrupted by printing deterioration and/or inherent noise of the devices, is considered from the viewpoint of statistical decision theory. The optimization consists of minimizing the expected risk for a weight function which is preassigned to measure the consequences of system decisions. As an alternative, minimization of the error rate for a given rejection rate is used as the criterion. The optimum recognition is thus obtained.

The optimum system consists of a conditional-probability densities computer; character channels, one for each character; a rejection channel; and a comparison network. Its precise structure and and ultimate performance depend essentially upon the signals and [illegible] structure.

Explicit examples for an additive Gaussian noise and a "cosine" noise are presented. Finally, an error-free recognition system and a possible criterion to measure the character style and deterioration are presented.

* Manuscript received by the PGEC, June 3, 1957.
† Burroughs, Corp., Paoli, Pa.

Introduction

CHARACTER recognition has been receiving considerable attention as the result of the phenomenal growth of office automation and the need for translating human language into machine language.[1,2] Broadly speaking, the character printed in conventional form and size on the document (checks, etc.) is first converted to electrical signals, and sufficient information is then extracted from the latter. The purpose of the recognition system is based on the observed data and on *a priori* knowledge of the signal and noise structure

[1] K. R. Eldredge, F. J. Kamphoefner, and P. H. Wendt, "Automatic input for business data processing system," *Proc. Eastern Joint Computer Conf.*, pp. 69–73; December 11, 1956.

[2] E. C. Greanias and Y. M. Hill, "Considerations in the design of character recognition devices," 1957 IRE National Convention Record, pt 4, vol. 5 pp. 119–126.

to identify which of the possible characters is present, or to reject if the data are ambiguous.

The over-all performance of the recognition system depends not only upon itself, but also upon the number of characters to be recognized, the character style, and noise statistics. In this paper the character style and noise statistics are assumed given and adequate, and the purpose of the paper is to obtain an optimum recognition system. For convenience, the recognition problem is considered one of statistical inference, so that useful results in decision theory can be applied.[3–5] To accomplish this, the notion of risk is employed and proper weights are assigned to various types of error, rejection, and correct recognition to measure the consequences of decisions. This results in an optimum system which minimizes the expected (average) risk function and includes a possible alternate system with a minimum error rate. The results reveal the explicit structure of an optimum system which is determined by the *a priori* noise statistics, the signal structure, and the preassigned weights.

System Approach to the Problem

One practical application of a character recognition system for business documents is to read arabic numerals and selected symbols printed in magnetic ink. A method[1] for achieving this is shown in Fig. 1. The character is first passed through the field of a permanent magnet where it is magnetized in a given direction before being scanned by the read head. From the read head, the printed character is converted into an electrical signal corresponding to the differentiation of the plane area of the character. The function of the recognition system is to examine the amplitude-time signal obtained by the read head and to decide which of the possible characters is being recognized.

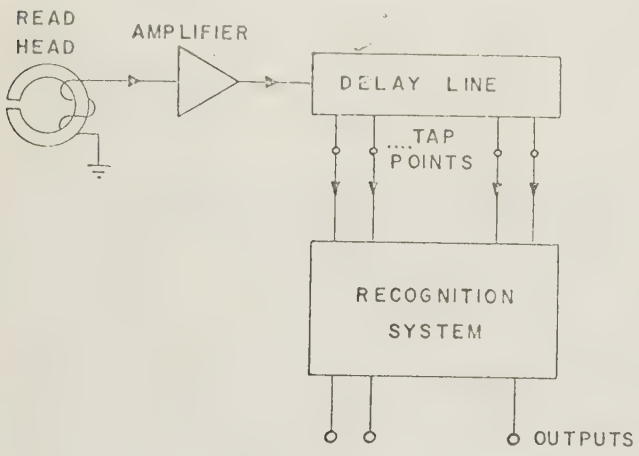


Fig. 1—A recognition system.

It is convenient, at times, to deal with the sampled data rather than the continuous time waveforms. By the sampling theory, if the number of samples is sufficiently large, little information carried by the continuous signal is lost. As shown in Fig. 1, the signal from the read head is first amplified and then fed into a tapped delay line. This serves as a means for sampling and acts as a temporary storage device to convert the series information into parallel information. Although not essential, sampled data are used in the following discussion.

Let the vector $v = (v_1, v_2, \cdots v_s)$ (subscript s being the number of samples) denote voltages on the taps of the delay line at the instant of sampling. (See Appendix I for the meaning of the Symbols.) The vector $a_i = (a_{i1}, a_{i2}, \cdots, a_{is})$ denotes the true sampled signal associated with the ith character where $i = 1, 2, \cdots, c$, c being the number of possible characters to be recognized. The vector v constitutes the input to the recognition system. It is assumed that the characters are distinct, *i.e.*, all a_i's are different.

In a simple form, the recognition system may consist of c separate channels, one for each character. Each channel obtains a weighted sum of v_i's, with properly chosen weights, b_{ij}. The output of the ith channel is

$$X_i(v) = \sum_{j=1}^{s} b_{ij} v_j. \tag{1}$$

This operation may be realized by a summing amplifier and possibly with some inverters to provide negative weights, if required. One possible set of weights is:

$$b_{ij} = \frac{a_{ij}}{\left[\sum_{j=1}^{s} a_{ij}^2\right]^{1/2}}. \tag{2}$$

The recognition system is known as a correlation network when the weights are defined by (2).

If the printing is perfect, and the reading devices are noiseless, the observed data v will be identical to one of the a_i's and therefore, it can easily be shown that the right channel of the correlation network has the largest (algebraic) output. Consequently, the recognition system identifies the character with absolute accuracy by the channel having the highest output. However, in practice, there are always, to some degree, deteriorations in printing and inherent noise in the devices. Therefore, the observed data v generally will not be identical to any of the a_i's. In view of this, ambiguities arise which may result in possible misrecognition. To safeguard against the occurrence of error, the recognition system should have provisions for examining the degree of ambiguity and making rejects when required. This function can be achieved in various ways; *e.g.*, whenever the next highest output of the correlation network exceeds some preassigned fraction of the highest output, the system will reject, otherwise the system

[3] A. Wald, "Statistical Decision Functions," John Wiley & Sons, Inc., New York, N. Y., 1950.

[4] D. Van Meter and D. Middleton, "Modern statistical approaches to reception in communication theory," IRE Trans., vol. PGIT-4, pp. 119–145; September, 1954.

[5] D. Middleton and D. Van Meter, "On optimum multiple-alternative detection of signals in noise," IRE Trans., vol. IT-1, pp. 1–9; September, 1955.

identifies the character by the channel having the highest output.

The system described above merely represents one of many possible recognition systems and is not necessarily optimum. A basic problem in the design of recognition systems is to evaluate the system performance in the presence of printing deterioration and inherent noise and to obtain an optimum system. Optimum performance depends primarily upon the character style and permissible deterioration. Greanias and Hill in a recent paper[2] describe the effects of character style and printing deterioration on the character recognition problem from the viewpoint of matching the character with an ideal character and further propose definitions for character quality and style factors. In this paper, the discussion is confined to the problem of obtaining an optimum recognition system for a given set of adequately styled characters and known statistics of character deterioration. The recognition problem is considered to be that of testing multiple hypotheses in the statistical inference. Consequently, the design and evaluation of a recognition system is comparable to a statistical test. Results of decision theory can be applied.[3–5].

In order to judge the relative merit of recognition systems, some criterion of evaluating system performance must be established. The error rate of the system for a given rejection rate is used as the performance criterion for cases where no distinction is made among misrecognitions. Cases may arise where different misrecognitions have different consequences; *e.g.*, the registering of a four as an asterisk may not be as serious an error as registering it as a nine. The criterion of minimum error rate is then no longer appropriate. Instead, the criterion of minimum risk[3] is employed. Proper weights are assigned to measure the consequences of errors, rejections, and correct recognitions. These weights indicate the loss incurred by the system for every possible decision. The loss, which should be regarded as negative utility, may actually represent loss in dollars or unit of utility in measuring the consequence of the system decision. The over-all performance of the system is judged by its expected (or average) risk.

In the following discussion, an optimum system which minimizes the expected risk is derived, and a system having minimum error rate is obtained. Examples are presented for illustration purposes. An error-free system and a possible criterion for judging character style and deterioration are also presented.

The Expected Risk

The vector $a_i = (a_{i1}, a_{i2}, \cdots, a_{is})$ in the s-dimensional space denotes the true sampled signal associated with the ith character ($i = 1, 2, \cdots, c$), where c and s are respectively, the number of possible characters to be recognized and the number of samples. Let $p = (p_1, p_2, \cdots, p_c)$ be the *a priori* distribution of characters (p_i is the *a priori* probability that the ith character occurs). Then, evidently,

$$\sum_{i=1}^{c} p_i = 1, \qquad p_i > 0. \tag{3}$$

The received data are denoted by a s-components vector $v = (v_1, v_2, \cdots, v_s)$. It is the signal corrupted by factors such as the deterioration of printing and inherent noise of the devices. *A priori* noise statistics and the manner in which various signals and noise are combined determine precisely the conditional probability density $F(v|a_i)$ of the observed data v when a_i is the incoming signal.

The space of decisions available to the recognition system consists of $c+1$ possible decisions $d_0, d_1, d_2, \cdots, d_c$. The quantity $d_i (i \neq 0)$ is the decision that the ith character is present while d_0 is the decision for reject. A basic problem in statistical decision theory is the selection of a proper decision rule δ. The rule is expressed as a vector function of the data v, namely, $\delta(v) = (\delta(d_0|v), \delta(d_1|v), \delta(d_2|v) \cdots \delta(d_c|v))$ with $c+1$ components, and satisfies the restriction that:

$$\sum_{i=0}^{c} \delta(d_i|v) = 1 \qquad \text{for all } v, \tag{4}$$

and

$$\delta(d_i|v) \geq 0 \qquad \text{for all } i \text{ and } v. \tag{5}$$

The quantity $\delta(d_i|v)$ is the probability that, for a given observed data v, the decision d_i will be made.

In order to judge the relative merits of the decision rules it is necessary to assign the weight function $W(a_i, d_j)$. This is a function of a_i and d_j, which is the loss incurred by the system if the decision d_j is made when a_i is the true signal. This measure of consequence for various d_j under various a_i is a datum of the problem and is given in advance. Let the weight function be:

$$W(a_i, d_j) = w_{ij} \qquad \begin{matrix} i = 1, 2 \cdots c \\ j = 0, 1, 2 \cdots c, \end{matrix} \tag{6}$$

where $w_{ii} (i \neq 0)$ is the weight of correct recognition of the ith character; $w_{ij} (i \neq j \neq 0)$ is the weight of misreading the ith character as the jth one, and $w_{i0} (i \neq 0)$ is the weight of rejecting the ith characters. Therefore, it is required that

$$w_{ij} > w_{i0} > w_{ii} \qquad (i \neq j \neq 0). \tag{7}$$

Usually, w_{ij} is much larger than w_{i0} since the most serious consideration in design of a character recognition system is the occurrence of undetected errors.

In general, w_{ij}'s may all differ, so that various misrecognitions, rejections, and correct recognitions can be properly weighted. The expected risk for any decision rule δ is

$$R(p, \delta) = \sum_{i=1}^{c} \sum_{j=0}^{c} \int_V \delta(d_j|v) p_i w_{ij} F(v|a_i) dv, \tag{8}$$

with integration over the entire observation space V.

The Minimum Risk System

The problem is then to choose a decision rule to minimize the average risk. By using (4), and since $\int_V F(v|a_i)dv = 1$ for all i, (8) may be written as:

$$R(p, \delta) = R_0(p) + R_1(p, \delta), \tag{9}$$

where

$$R_0 = \sum_{i=1}^{c} p_i w_{i0}, \tag{10}$$

$$R_1 = \int_V \sum_{j=0}^{c} \delta(d_j | v) Z_j(v) dv, \tag{11}$$

and

$$Z_j(v) = \begin{cases} \sum_{i=1}^{c} (w_{ij} - w_{i0}) p_i F(v | a_i); & j = 1, 2 \cdots c \\ 0 & \text{for } j = 0. \end{cases} \tag{12}$$

The symbol R_0 will express the expected risk when rejection is made for all recognition and R_1 is that part of R which may be adjusted through the choice of δ. Evidently

$$R_1(p, \delta) \geq \int_V \min_j [Z_j(v)] dv, \tag{13}$$

and the equality sign holds if, and only if, the decision rule is chosen as:

$$\begin{aligned} &\delta^*(d_k | v) = 1, \\ &\delta^*(d_j | v) = 0 \qquad \text{for all } j \neq k \end{aligned} \tag{14}$$

whenever

$$\min_j [Z_j(v)] = Z_k(v). \tag{15}$$

This is the optimum decision rule δ^* (the Bayes strategy) which minimizes the expected risk and is nonrandomized since its components are either zero or one. Therefore, R_1 for this decision rule is always nonpositive, and its expected risk (the Bayes risk) is no greater than R_0. The expected risk for the optimum decision rule, δ^*, is

$$R(p, \delta^*) = \sum_{i=1}^{c} p_i w_{i0} + \int_V \min_j [Z_j(v)] dv. \tag{16}$$

Eqs. (14) and (15) reveal that the optimum system for character recognition consists of a computer which evaluates $F(v|a_i)$'s; $(i = 1, 2, \cdots, c)$ for an observed data v; computes the various $Z_j(v)$ $(j = 1, 2, \cdots, c)$; examines and compares these $Z_j(v)$ $(j = 0, 1, 2, \cdots, c)$; selects the smallest (algebraically) one, say $Z_k(v)$; and finally makes the decision d_k [having the same subscript as $Z_k(v)$]. Of course, this method of setting up the computing procedures is not unique; *e.g.*, any ordering-preserving transformation may be used. In any event, the system must be equivalent to the above.

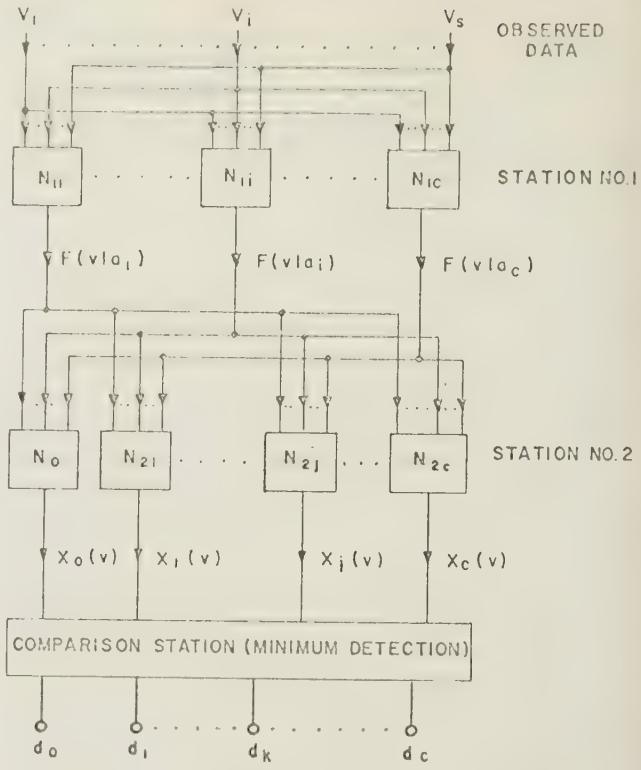


Fig. 2—Functional diagram of an optimum system.

The functional diagram of the optimum system is shown in Fig. 2. Station No. 1 consists of c similar component networks. Each network receives the observed data $v = (v_1, v_2, \cdots, v_s)$ and computes the corresponding conditional probability density $F(v|a_i)$ as its output. This operation depends only upon the *a priori* knowledge of signal and noise structure and on the observed data v; it does not depend upon the weight function $W(a_i, d_j)$ or on *a priori* probability distribution of signals, p.

The outputs of station No. 1 are fed to station No. 2, which consists of c character channels $N_{2j}(j \neq 0)$, and one rejection channel, N_0. They perform the linear operation of weighting each input and then the summing of all weighted inputs. The weights are $p_i w_{ij}$'s and $p_i w_{i0}$'s. The output of the rejection channel is

$$X_0(v) = \sum_{i=1}^{c} w_{i0} p_i F(v | a_i), \tag{17}$$

while the outputs of character channels are

$$X_j(v) = \sum_{i=1}^{c} w_{ij} p_i F(v | a_i) \quad (j = 1, 2 \cdots c). \tag{18}$$

The comparison station receives X's from station No. 2, examines all its inputs, and makes decision by selecting the algebraically smallest of the $c+1$ X's. If the rejection channel has the smallest one, the system rejects. If one of the character channels has the smallest output (say $X_k(v)$, $(k \neq 0)$) then the system recognizes the signal as the kth character.

Since $X_j(v)$ $(j = 0, 1, \cdots, c)$ is equal to $Z_j(v) + X_0(v)$,

this system makes decisions in accordance with δ^* as defined by (14) and (15), and thus minimizing the expected risk.

Probabilities of Error and Rejection

The expected risk provides a means for evaluating the performance of a recognition system. At times, it may be desirable to compute the probabilities of error, rejection, and correct recognition as an auxiliary set of merit figures. They are obtained for any decision rule δ as follows:

The probability of correct recognition is:

$$P_c(\delta) = \int_V \sum_{i=1}^{c} \delta(d_i \mid v) p_i F(v \mid a_i) dv; \tag{19}$$

the probability of rejection, or rejection rate, is:

$$P_r(\delta) = \int_V \delta(d_0 \mid v) \sum_{i=1}^{c} p_i F(v \mid a_i) dv; \tag{20}$$

and the probability of misrecognition, or error rate, is:

$$P_e(\delta) = 1 - P_c - P_r. \tag{21}$$

Eqs. (19)–(21) result directly from the fact that $\int_V \delta(d_0 \mid v) F(v \mid a_i) dv$ and $\int_V \delta(d_i \mid v) F(v \mid a_i) dv$ are, respectively, the conditional probabilities of rejection of the ith character and correct recognition of the ith character.

Criterion of Minimum Error Rate

Cases may arise in which the criterion of judging the system performance is the magnitude of its error rate for a given rejection rate. In using this criterion, the optimum recognition system is the one which, for a given rejection rate, α, has a minimum error rate. The optimum decision rule is obtained as: (See Appendix II for proof.)

$$\delta^{**}(d_k \mid v) = 1 \qquad (k \neq 0) \tag{22}$$

whenever

$$p_k F(v \mid a_k) \geq p_j F(v \mid a_j) \quad \text{for all } j \neq k, \text{ and}$$

$$p_k F(v \mid a_k) \geq \beta \sum_{i=1}^{c} p_i F(v \mid a_i), \tag{23}$$

and

$$\delta^{**}(d_0 \mid v) = 1, \tag{24}$$

whenever

$$\beta \sum_{i=1}^{c} p_i F(v \mid a_i) > p_j F(v \mid a_j) \quad \text{for all } j (j = 1, 2 \cdots c), \tag{25}$$

where $\beta (0 \leq \beta \leq 1)$ is a nonnegative constant determined by the condition that $P_r(\delta^{**}) = \alpha$; namely,

$$\int_V \delta^{**}(d_0 \mid v) \sum_{i=1}^{c} p_i F(v \mid a_i) dv = \alpha. \tag{26}$$

The constant β increases with increasing α, and $P_e(\delta^{**})$ and $P_c(\delta^{**})$ are monotonic decreasing functions of α.

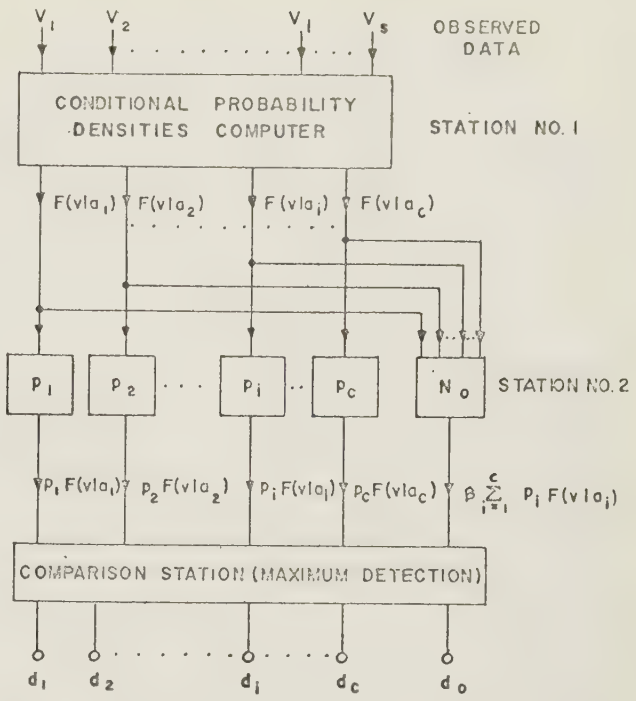


Fig. 3—Functional diagram of the system having the minimum error rate.

The change in the constant β provides a control over the error-reject ratio.

The optimum rule δ^{**} provides the basis for the functional diagram of the system of minimum error rate as shown in Fig. 3. The first station is identical to that of the minimum risk system (see Fig. 2) which computes the conditional probability densities $F(v \mid a_i)$'s.

The second station for this system is somewhat similar to that shown in the functional diagram for the minimum risk system (see Fig. 2). The c-character channels perform the weighting operation and have $p_i F(v \mid a_i)$ as outputs. The rejection channel, N_0, performs the operation of weighting and summing and has $\beta \sum_{i=1}^{c} p_i F(v \mid a_i)$ as its output. All of these $c+1$ outputs are nonnegative. The comparison station then examines these outputs and selects the largest. If the output of the rejection channel is the largest, the system rejects; otherwise the system will identify the character by the channel having the largest output.

It can be shown that the system depicted in Fig. 2 reduces to the system shown in Fig. 3 when β is replaced by $(w_m - w_r)/(w_m - w_c)$, and the following weight function is used.

$$W(a_i, d_j) = \begin{cases} w_c & \text{for } i = j \neq 0 \\ w_r & \text{for } i \neq 0, j = 0. \\ w_m & \text{for } i \neq j \neq 0. \end{cases} \tag{27}$$

Examples

1) As an illustration, consider a condition where the signals and noise are additive, and the noise has independent normal distribution. To be explicit, the probability density function of the noise of the jth component of the ith character is taken as:

$$\frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left\{-\frac{(v_j - a_{ij})^2}{2\sigma_{ij}^2}\right\}, \tag{28}$$

where $v_j - a_{ij}$ is the noise and σ_{ij}^2 is the given variance.

The conditional probability density $F(v|a_i)$, under the assumption that noise is statistically independent, is

$$F(v|a_i) = \frac{\exp\left\{-\sum_{j=1}^{s} \frac{(v_j - a_{ij})^2}{2\sigma_{ij}^2}\right\}}{(2\pi)^{s/2} \prod_{j=1}^{s} \sigma_{ij}}. \tag{29}$$

Therefore, the last expression dictates the precise structure of station No. 1 for the optimum system. Each component circuit performs the operations of taking differences, squaring, weighting, summing, and taking exponential. This station is common for a minimum risk or minimum error rate system. The structures of the second station and comparison station are indicated in Figs. 2 and 3.

2) In this example, the signal and noise structure are such that the conditional probability density for a given length, $|v|$, of v is directly proportional to the cosine of the angle θ between vectors v and a_i for $|\theta| < \pi/2$ and is zero elsewhere, and that the distributions of $|v|$ for given a_i are identical for all i.[6] [It is denoted as $f(|v|)$.] In other words, $F(v|a_i)$ can be written as

$$\begin{aligned} F(v|a_i) &= \rho f(|v|) \frac{a_i \cdot v}{|a_i||v|} && \text{for } a_i \cdot v > 0 \\ &= 0 && \text{elsewhere,} \end{aligned} \tag{30}$$

where ρ is a constant independent of i and is determined by the fact that $\int_V F(v|a_i)dv = 1$, and $a_i \cdot v$ denotes the scalar product of vectors a_i and v.

An inspection of the optimum decision rule (δ^* or δ^{**}) reveals that the system remains optimum if the first station is to compute $T[F(v|a_i)]$ instead of $F(v|a_i)$, where T is defined as

$$\begin{aligned} T(v|a_i) &= \frac{|v|}{\rho f(|v|)} F(v|a_i) \\ &= \begin{cases} \dfrac{a_i}{|a_i|} \cdot v = \sum_{j=1}^{s} b_{ij} v_j & \text{for } a_i \cdot v > 0 \\ 0 & \text{for } a_i \cdot v \leq 0, \end{cases} \end{aligned} \tag{31}$$

where b_{ij}'s are constants [see (2)]. This operation can be easily realized. Each component of the first station is simply a correlation network followed by a half-wave rectifying circuit. The circuit passes the positive output of the correlation network unaltered and converts its negative output into zero.

The above results also indicate that the recognition system described in the second section of this paper is not optimum for the particular signal and noise structure as given in examples 1 or 2.

[6] This particular signal and noise structure was suggested by the author's colleague, I. M. Sheaffer, Jr., Burroughs Corp., Paoli, Pa.

Error-Free System

For convenience, let V_i denote the set (or region) of all possible observation v when the ith character is present, and let $\tilde{V}_i$ be the largest subset of V_i so that $\tilde{V}_i$'s are nonoverlapping. If noise distributions are so truncated and the signal vectors a_i's are so placed that all $\tilde{V}_i$'s are nonempty, then an error-free system for character recognition does exist. Evidently for any observed data v belonging to $\tilde{V}_k$, only $F(v|a_k)$ is nonzero while all others are zero; the character can then be identified with certainty. On the other hand, if the data v do not belong to any one of the $\tilde{V}_i$'s, then more than one of the $F(v|a_i)$'s will be nonzero. This results in the data being ambiguous for recognition purpose, and an error-free system will reject. Symbolically, the error-free decision rule is:

$$\begin{aligned} &\delta(d_k|v) = 1 \quad \text{if } F(v|a_k) > 0 \\ &\text{and} \\ &F(v|a_i) = 0 \quad \text{for all } i \neq k, \\ &\text{and} \\ &\delta(d_0|v) = 1, \quad \text{otherwise,} \end{aligned} \tag{32}$$

the rejection rate is determined by the probability measures of $\tilde{V}_i$'s, namely $\int_{\tilde{V}_i} F(v|a_i)dv$. The latter is determined by the character style and allowable deterioration. The character style may be considered ideal and the control over the printing perfect, if the resultant $\int_{\tilde{V}_i} F(v|a_i)dv$ is unity for all i, and all characters with allowable deterioration can then be recognized with neither an error nor reject. In this sense, the probability measures of $\tilde{V}_i$'s may be used to evaluate the combined quality of the character style and printing.

Conclusion

The decision theory has been successfully applied to the problem of character recognition. By employing the concept of risk, differences in consequences for various decisions have been taken into consideration. A rejection channel has been introduced to examine the degree of ambiguity of input signal and make rejections when necessary.

As developed, the structure and performance of an optimum system depend upon the signal and noise statistics; therefore, *a priori* knowledge of these statistics is required. Usually, a realistic estimate of noise statistics is not easy to obtain. However, it is sincerely felt that the requirement for high grade performance in character recognition warrants the expenditures in this direction.

Quite often an optimum system may prove to be too expensive for mechanization. Nevertheless, the results presented in this paper are considered useful in that they provide insight into the recognition problem and furnish an ideal system, which actual recognition circuitry may be patterned after.

Although it is recognized as being beyond the scope of this paper, it is worth mentioning that one practical

pproach to the over-all problem would be to design dequately the character style and to control properly he printing process so that a reliable system would not e too far fetched or difficult to ultimately realize.

Appendix I.

List of Symbols

$a_i = (a_{i1}, a_{i2}, \cdots, a_{is})$, s-dimensional vector associated with the ith character, $(i = 1, 2, \cdots, c)$.
$a_{ij} = j$th sample of the signal of the ith character.
$|a_i| = (\sum_{j=1}^{s} a_{ij}^2)^{1/2}$, the length of vector a_i.
c = number of characters.
d_0 = decision that rejection be made.
d_j = decision that the signal is the jth character $(j = 1, 2, \cdots, c)$.
$f(|v|)$ = probability density of the length of v.
$F(v|a_i)$ = conditional probability density for the observed data v when a_i is the incoming signal.
i, j, k = indexes.
N_{1i} = a network of station No. 1, $i = 1, 2, \cdots, c$.
N_{2i} = a network of station No. 2, $i = 1, 2, \cdots, c$.
N_0 = the network of rejection channel.
P_c = probability of correct recognition.
P_r = probability of rejection (rejection rate).
P_e = probability of misrecognition (error rate).
p_i = *a priori* probability that the ith character occurs, $(i = 1, 2, \cdots, c)$.
$p = (p_1, p_2, \cdots, p_c)$.
$R(p, \delta)$ = expected risk of the system; $R = R_0 + R_1$.
$R_0(p)$ = expected risk of the system when rejection is made for all recognition.
$R_1(p, \delta)$ = part of R which is dependent upon δ.
s = number of samples.
T = functional transformation.
V = s-dimension observation space.
V_i = set of all v when the ith character is present.
$\tilde{V}_i$ = largest subset of V_i such that $\tilde{V}_i \cap \tilde{V}_j = 0$ for all $j \neq i$.
$v = (v_1, v_2, \cdots, v_s)$, a vector in V.
$|v| = (\sum_{i=1}^{s} v_i^2)^{1/2}$, the length of vector v.
vi = ith component of the observed data v.
$W(a_i, d_j)$ = weight function.
$w_{ij} = W(a_i, d_j)$.
w_c, w_r, w_m = weights.
$x_i(v)$ = output of the ith channel.
α = permissible rejection rate.
β = constant.
$\delta(v)$ = decision rule, $\delta = [\delta(d_0|v), \delta(d_1|v) \cdots \delta(d_c|v)]$.
$\delta^*(v)$ = optimum decision rule which minimizes the expected risk.
$\delta^{**}(v)$ = optimum decision rule which minimizes the error rate.
θ = angle.
ρ = a normalizing constant.
σ_{ij}^2 = statistical variance of noise.

Appendix II.

To Prove That δ^{**} Has a Minimum Error Rate for a Given Rejection Rate

Without loss of generality, it is assumed that the absolute probability density of the occurrence of v, namely, $\sum_{i=1}^{c} p_i F(v|a_i)$ is nonzero over the entire observation space V. Otherwise, the set over which $\sum_{i=1}^{c} p_i F(v|a_i)$ is zero is first deleted.

Let $m(v)$ be the subscript such that

$$\max_i [p_i F(v|a_i)] = p_m F(v|a_m) \tag{33}$$

and let $\delta^1(v)$ be any arbitrary decision rule having the same rejection rate as δ^{**}. It is to be proved that $P_e(\delta^1) \geq P_e(\delta^{**})$.

For every $\delta^1(v)$, a decision rule $\delta^2(v)$ can be constructed as follows:

For every v,

$$\begin{aligned} \delta^2(d_0|v) &= \delta^1(d_0|v) \\ \delta^2(d_m|v) &= 1 - \delta^1(d_0|v) = \sum_{i=1}^{c} \delta^1(d_i|v) \\ \delta^2(d_i|v) &= 0 \quad \text{for all } i \neq 0 \neq m. \end{aligned} \tag{34}$$

Evidently,

$$P_r(\delta^2) = P_r(\delta^1) = \alpha, \tag{35}$$

and

$$\begin{aligned} P_e(\delta^1) &= \int_V \sum_{i=1}^{c} \delta^1(d_i|v) p_i F(v|a_i) dv \\ &\leq \int_V \sum_{i=1}^{c} \delta^1(d_i|v) p_m F(v|a_m) dv \\ &= \int_V \delta^2(d_m|v) p_m F(v|a_m) dv \\ &= P_e(\delta^2). \end{aligned} \tag{36}$$

It follows from (35) and (36) that

$$P_e(\delta^2) \leq P_e(\delta^1). \tag{37}$$

That is, δ^2 is better than δ^1 (or at least as good) in the sense that for the same rejection rate δ^2 has an error rate smaller than, or equal to, that of δ^1.

The next step is to show that $P_e(\delta^{**}) \leq P_e(\delta^2)$. As shown in (22) ad (23), the decision rule δ^{**} partitions the observation space V into two nonintersecting regions, V_0^{**} and $V - V_0^{**}$, so that for every $v \in V_0^{**}$

$$p_m F(v|a_m) < \beta \sum_{i=1}^{c} p_i F(v|a_i) \tag{38a}$$

$$\delta^{**}(d_0|v) = 1, \tag{38b}$$

and for every $v \in V - V_0^{**}$

$$p_m F(v \mid a_m) \geq \beta \sum_{i=1}^{c} p_i F(v \mid a_i) \tag{39a}$$

$$\delta^{**}(d_m \mid v) = 1. \tag{39b}$$

Let V_0^2 be the largest subspace of V such that $\delta^2(d_0 \mid v)$ is nonzero for all v belonging to V_0^2. V_0^2 is not properly contained in V_0^{**}. This follows readily from the condition that $P_r(\delta^{**}) = P_r(\delta^2)$. The latter may be written as:

$$\int_{V_0^{**} \; V_0^{**} \cap V_0^2} \sum_{i=1}^{c} p_i F(v \mid a_i) dv + \int_{V_0^{**} \cap V_0^2} [1 - \delta^2(d_0 \mid v)] \sum_{i=1}^{c} p_i F(v \mid a_i) dv = \int_{V_0^2 - V_0^{**} \cap V_0^2} \delta^2(d_0 \mid v) \sum_{i=1}^{c} p_i F(v \mid a_i) dv. \tag{40}$$

Substitution of (38) and (39) in (40) gives:

$$\int_{V_0^{**} - V_0^{**} \cap V_0^2} p_m F(v \mid a_m) dv + \int_{V_0^{**} \cap V_0^2} [1 - \delta^2(d_0 \mid v)] p_m F(v \mid a_m) dv \leq \int_{V_0^2 - V_0^{**} \cap V_0^2} \delta^2(d_0 \mid v) p_m F(v \mid a_m) dv. \tag{41}$$

The equality sign prevails if, and only if, $p_m F(v \mid a_m)$ is equal to $\beta \sum_{i=1}^{c} p_i F(v \mid a_i)$ throughout the region $V_0^{**} \cup V_0^2$.

The probabilities of correct recognition of δ^{**} and δ^2 may be written respectively as:

$$P_c(\delta^{**}) = \int_{V - V_0^{**}} p_m F(v \mid a_m) dv = \int_{V - (V_0^{**} \cup V_0^2)} p_m F(v \mid a_m) dv + \int_{V_0^2 - V_0^{**} \cap V_0^2} p_m F(v \mid a_m) dv, \tag{42a}$$

$$P_c(\delta^2) = \int_{V} \sum_{i=1}^{c} \delta^2(d_i \mid v) p_i F(v \mid a_i) dv = \int_{V - (V_0^{**} \cup V_0^2)} p_m F(v \mid a_m) dv + \int_{V_0^{**} - V_0^{**} \cap V_0^2} p_m F(v \mid a_m) dv + \int_{V_0^{**} \cap V_0^2} \delta^2(d_m \mid v) p_m F(v \mid a_m) dv + \int_{V_0^2 - V_0^{**} \cap V_0^2} \delta^2(d_m \mid v) p_m F(v \mid a_m) dv. \tag{42b}$$

In accordance with (42), (41) is equivalent to $P_e(\delta^{**}) \leq P_e(\delta^2)$. Proof that $P_e(\delta^{**}) \leq \delta P_e(\delta^1)$ is thus completed.

Acknowledgment

This paper is based upon research performed in the Electromechanisms Department of the Burroughs Corporation Research Center at Paoli, Pa. The author wishes to thank his colleagues and Department Manager, G. A. Baird for their encouragement and assistance relative to this paper. The permission of the Burroughs Corporation for release of the material presented here is also acknowledged.

An Analysis of Certain Errors in Electronic Differential Analyzers

I—Bandwidth Limitations*

PAUL C. DOW, JR.†

Summary—When a differential analyzer or analog computer is set up to solve a given equation, certain errors are introduced into the solution because the components of the computer are not perfect. Stated another way, the computer produces the solution to an equation, called the "machine equation," which differs from the given equation. In this paper the computer imperfections considered are 1) operational amplifier frequency response, 2) capacitor leakage resistance, and 3) stray capacitance in summing and integrating amplifier circuits. It is shown that when the given equation is a system of one or more linear differential equations with constant coefficients, the machine equation can be expressed approximately as an equation of the same degree as the given equation and with constant coefficients which are functions of the coefficients of the given equation and of the computer imperfections.

Introduction

IT has been shown[1–3] that bandwidth limitations inherent in summers and integrators used in differential analyzers produce shifts in the characteristic roots of the differential equation being solved. The methods of analysis used to obtain these shifts depend for their simplicity on particular computer setups. In this paper no restriction as to the method of computer setup is imposed. Furthermore, the purpose here will be to find not the errors in the roots of the equation but rather the errors in the constant coefficients of the differential equation, whether or not the roots are known.

The bandwidth limitations of summers and integrators have been considered by a number of authors[4–6] and are briefly reviewed here. The transfer functions of summers and integrators are developed and then used to find the exact equation solved by the computer. This in turn is reduced by certain approximations to an equation of the same order as the given equation.

* Manuscript received by the PGEC, June 19, 1957; revised manuscript received, September 17, 1957. This paper contains part of the results of a dissertation submitted in partial fulfillment of the requirements for the Ph.D. degree at the University of Michigan, Ann Arbor, Mich.

† Capt., USAF, Air Force Ballistic Missile Div., Air Res. and Dev. Command, Inglewood, Calif.

[1] F. H. Raymond, "Sur un type général de machines mathematique algebriques," *Ann. Telécommun.*, vol. 5, pp. 2–20; January, 1950.

[2] A. B. Macnee, "Some limitations on the accuracy of electronic differential analyzers," PROC. IRE, vol. 40, pp. 303–308; March, 1952.

[3] V. A. Marsocci, "An error analysis of electronic analog computers," IRE TRANS., vol. EC-5, pp. 207–213; December, 1956.

[4] G. A. Korn, "Design of dc electronic integrators," *Electronics*, vol. 21, pp. 124–126; May, 1948.

[5] C. A. A. Wass, "Introduction to Electronic Analogue Computers," McGraw-Hill Book Co., Inc., New York, N. Y.; 1955.

[6] L. Cahn, "Accuracy of an analog computer," IRE TRANS., vol. EC-2, pp. 12–18; December, 1953.

Operational Amplifier

Many computer amplifiers have frequency response characteristics as shown in Fig. 1. The open loop transfer function of such an amplifier is

$$G(s) = \frac{G_0}{T_0 s + 1} \cdot \tag{1}$$

If ω_0 is the frequency at which the open loop gain is unity, then $|G(s)| = 1$ when $s = j\omega_0$. From (1) one has, if $G_0 \gg 1$,

$$\omega_0 \cong \frac{G_0}{T_0} \cdot \tag{2}$$

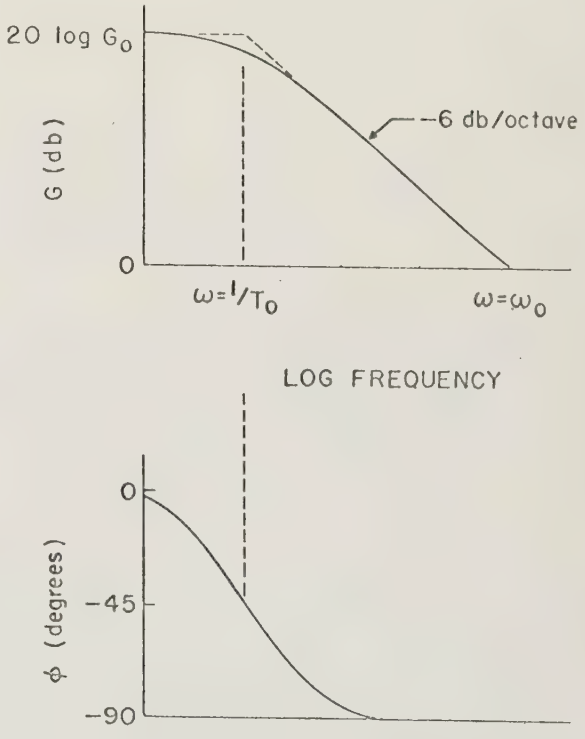


Fig. 1—Frequency response of open loop amplifier.

It is shown in the Appendix that even those amplifiers which closely approximate the frequency response characteristics of Fig. 1 at high frequencies but not at low frequencies can still be represented by a first-order transfer function such as given by (1) for the purpose of this analysis.

Summer

The amplifier connected as a summer is shown in Fig. 2, in which the stray capacitance and resistance to ground are shown in dotted lines.

If Z_i is the impedance between each input terminal and the amplifier summing junction P, and Z_f is the feedback impedance, and there is no grid current or amplifier unbalance, one has

$$\sum_i \frac{V_i - V}{Z_i} = \frac{V - V_0}{Z_f} + sC_sV + \frac{V}{R_s} \tag{3}$$

and

$$V_0 = -G(s)V. \tag{4}$$

Eliminating V from (3) and (4) yields

$$V_0 = \frac{-\sum_i \frac{Z_f}{Z_i} V_i}{1 + \frac{1}{G(s)}\left[1 + sC_sZ_f + \sum_i \frac{Z_f}{Z_i} + \frac{Z_f}{R_s}\right]}. \tag{5}$$

From Fig. 2 it is seen that

$$Z_i = \frac{R_i}{1 + sR_iC_i} \tag{6}$$

$$Z_f = \frac{R_f}{1 + sR_fC_f}. \tag{7}$$

Substituting (1), (6), and (7) in (5) yields

$$V_0 = \frac{-\sum_i \frac{R_f(1 + sR_iC_i)}{R_i(1 + sR_fC_f)} V_i}{1 + \frac{T_0s + 1}{G_0}\left[1 + \frac{sR_fC_s}{1 + sR_fC_f} + \sum_i \frac{R_f(1 + sR_iC_i)}{R_i(1 + sR_fC_f)} + \frac{R_f}{R_s(1 + sR_fC_f)}\right]}. \tag{8}$$

For economy of notation, let R_iC_i for each i be the same. (If this assumption does not apply, the following equations become more lengthy but the same method of analysis can be used.) Then (8) becomes

$$V_0 = \frac{-(1 + sR_iC_i)\sum_i \frac{R_f}{R_i} V_i}{\alpha + \beta s + \gamma s^2} \tag{9}$$

where

$$\alpha = 1 + \frac{1}{G_0}\left(1 + \sum_i \frac{R_f}{R_i} + \frac{R_f}{R_s}\right)$$

$$\beta = R_fC_f + \frac{T_0}{G_0}\left(1 + \sum_i \frac{R_f}{R_i} + \frac{R_f}{R_s}\right) + \frac{1}{G_0}\left(R_fC_f + R_fC_s + R_iC_i \sum_i \frac{R_f}{R_i}\right)$$

$$\gamma = \frac{T_0}{G_0}\left(R_fC_f + R_fC_s + R_iC_i \sum_i \frac{R_f}{R_i}\right). \tag{10}$$

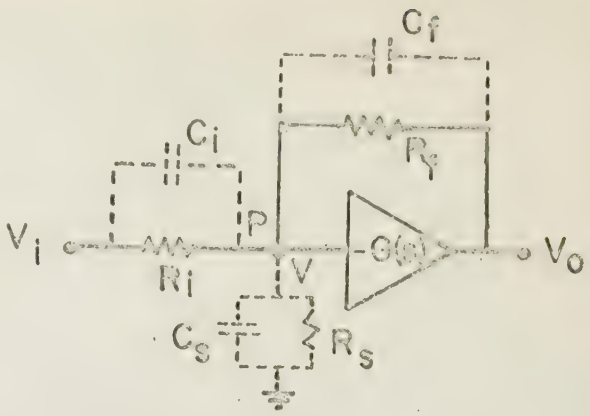


Fig. 2—Amplifier connected as a summer.

In most computer applications it will generally be true that

$$\alpha \cong 1$$

$$\beta \cong R_fC_f + \frac{T_0}{G_0}\left(1 + \sum_i \frac{R_f}{R_i} + \frac{R_f}{R_s}\right)$$

$$\gamma s^2 \ll 1 \tag{11}$$

and (9) can be written

$$V_0 = \frac{-(1 + sR_iC_i)\sum_i \frac{R_f}{R_i} V_i}{1 + \beta s}. \tag{12}$$

The final expression for the summer is written

$$V_0 = -\frac{T_is + 1}{T_ss + 1} \sum_i \frac{R_f}{R_i} V_i \tag{13}$$

where

$$T_i = R_iC_i \tag{14}$$

and

$$T_s = \frac{T_0}{G_0}\left(1 + \sum_i \frac{R_f}{R_i} + \frac{R_f}{R_s}\right) + R_fC_f. \tag{15}$$

Integrator

The amplifier connected as an integrator is shown in Fig. 3, in which the stray capacitance and leakage resistance are shown in dotted lines.[7] Eq. (9), which was obtained for the summer, will also apply for the inte-

[7] In a second article the effect of dielectric absorption and complex permittivity of the dielectric in integrator capacitors will be considered.

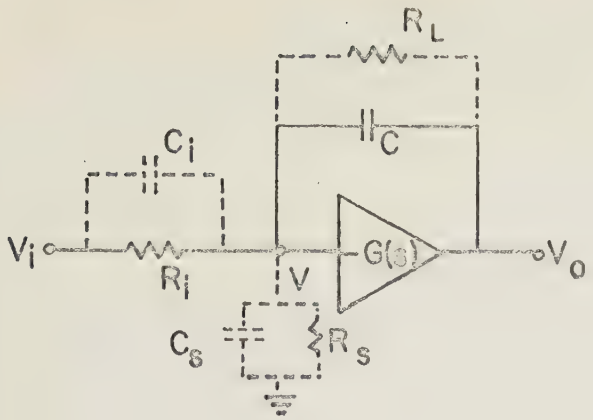


Fig. 3—Amplifier connected as an integrator.

grator if R_f and C_f are replaced by R_L and C, respectively. Then

$$V_0 = \frac{(1 + sR_iC_i)\sum_i \frac{V_i}{R_iC}}{\alpha' + \beta' s + \gamma' s^2} \tag{16}$$

where

$$\alpha' = \frac{1}{R_LC}\left(1 + \frac{1}{G_0}\right) + \frac{1}{G_0}\left(\sum_i \frac{1}{R_iC} + \frac{1}{R_sC}\right)$$

$$\beta' = 1 + \frac{T_0}{G_0}\left(\frac{1}{R_LC} + \sum_i \frac{1}{R_iC} + \frac{1}{R_sC}\right) + \frac{1}{G_0}\left(1 + \frac{C_s}{C} + R_iC_i\sum_i \frac{1}{R_iC}\right)$$

$$\gamma' = \frac{T_0}{G_0}\left(1 + \frac{C_s}{C} + R_iC_i\sum_i \frac{1}{R_iC}\right). \tag{17}$$

In most computer applications it will be true that (17) can be written approximately

$$\alpha' \cong \frac{1}{R_LC} + \frac{1}{G_0}\sum_i \frac{1}{R_iC}$$

$$\beta' \cong 1$$

$$\gamma' \cong \frac{T_0}{G_0} \tag{18}$$

and (16) can be written

$$V_0 = \frac{-(1 + sR_iC_i)\sum_i \frac{V_i}{R_iC}}{\left(\frac{T_0}{G_0}s + 1\right)\left(s + \frac{1}{R_LC} + \frac{1}{G_0}\sum_i \frac{1}{R_iC}\right)}. \tag{19}$$

The final expression for the integrator is written

$$V_0 = -\frac{T_1(T_is + 1)}{(T_1s + 1)(T_2s + 1)}\sum_i \frac{V_i}{R_iC} \tag{20}$$

where

$$\frac{1}{T_1} = \frac{1}{R_LC} + \frac{1}{G_0}\sum_i \frac{1}{R_iC}$$

$$T_2 = \frac{T_0}{G_0}$$

$$T_i = R_iC_i. \tag{21}$$

Solution of a Single Differential Equation

The equation to be solved, called hereafter the "given equation," is

$$\left(\sum_{k=0}^{m} a_k s^k\right) x = f(s). \tag{22}$$

Although a particular computer setup will be used to solve(22), it will readily be seen that the method will apply equally well to any computer setup. Fig. 4 shows the computer setup used in this analysis.

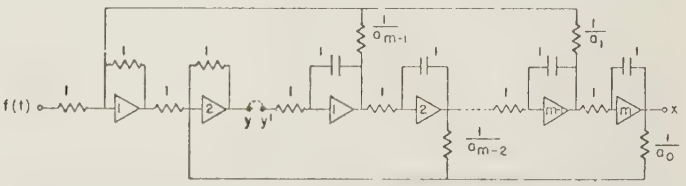


Fig. 4—Computer setup for linear differential equations.

To determine the equation which the computer is solving, the voltage at y is evaluated in terms of y', then y is set equal to y' by connecting the two points together. This yields

$$y = -y'\left[\frac{a_0T_1^m(T_is+1)^{m+1+\delta}}{(\delta T_{s1}s+1)(T_{s2}s+1)(T_1s+1)^m(T_2s+1)^m} + \cdots + \frac{a_{m-1}T_1(T_is+1)^{2+\delta}}{(\delta T_{s1}s+1)(T_{s2}s+1)(T_1s+1)(T_2s+1)}\right] + \frac{f(s)(T_is+1)^{1+\delta}}{(\delta T_{s1}s+1)(T_{s2}s+1)} \tag{23}$$

where T_1 and T_2 are the integrator time constants defined by (21) and are assumed the same for each integrator to keep the equations reasonably compact. T_{s1} and T_{s2} are the time constants of the first and second summers, respectively, defined by (15) and $T_i = R_iC_i$. $\delta = 1$ or 0 if the coefficient a_k of the term in which it appears is determined by a resistor connected to the input of the first or second summer, respectively, or in the last term if $f(t)$ is an input to the first or second summer, respectively.

Noting that

$$x = \frac{y'T_1^m(T_is + 1)^m}{(T_1s + 1)^m(T_2s + 1)^m} \tag{24}$$

(23) can be rewritten as

$$\left[\left(s+\frac{1}{T_1}\right)^m+\sum_{k=0}^{m-1}\frac{a_k\left(s+\frac{1}{T_1}\right)^k(T_is+1)^{m-k+1+\delta}}{(\delta T_{s1}s+1)(T_{s2}s+1)(T_2s+1)^{m-k}}\right]1$$
$$=\frac{f(s)(T_is+1)^{m+1+\delta}}{(\delta T_{s1}s+1)(T_{s2}s+1)(T_2s+1)^m}. \quad (25)$$

Eq. (25) is the exact equation solved by the analog computer setup shown in Fig. 4. It is seen that the characteristic equation is at most of order $2m+2$ in s. The given equation has a characteristic equation of order m in s. Thus $m+2$ extraneous roots are introduced by the m integrators and two summers. Macnee has shown[2] that there are m extraneous roots

$$s=-\frac{1}{T_2} \quad (26)$$

and two roots

$$s=-\frac{1}{T_{s1}}$$
$$s=-\frac{1}{T_{s2}}. \quad (27)$$

Since they are large and negative they will damp out quickly and have negligible effect on the solution. The remaining m roots, if the errors introduced by the imperfect components are small, are approximately equal to the m roots of the given equation.

If s_i $(i=1, \cdots, m)$ are the roots of the given characteristic equation and if

$$\frac{1}{T_1}\ll|s_i|\ll\frac{1}{T_{s1}},\frac{1}{T_{s2}},\frac{1}{T_2},\frac{1}{T_i}, \quad (28)$$

then the following approximations will be valid in those regions of the complex plane where $s\cong s_i$:

$$\frac{1}{T_{s1}s+1}\cong 1-T_{s1}s$$
$$\left(1+\frac{1}{T_1s}\right)^m\cong 1+\frac{m}{T_1s}$$
$$(1+T_{s1}s)(1+T_2s)\cong 1+(T_{s1}+T_2)s, \text{ etc.} \quad (29)$$

If these approximations are made, (25) becomes

$$\left[s^m\left(1+\frac{m}{T_1s}\right)+\sum_{k=0}^{m-1}s^ka_k\left\{1-\delta T_{s1}s-T_{s2}s-(m-k)T_2s+(m-k+1+\delta)T_is+\frac{k}{T_1s}\right\}\right]x$$
$$=\left\{1-\delta T_{s1}s-T_{s2}s-mT_2s+(m+1+\delta)T_is\right\}f(s). \quad (30)$$

This can be further simplified to

$$\sum_{k=0}^{m}s^k\left[a_k+a_{k+1}\frac{k+1}{T_1}-a_{k-1}\left\{\delta T_{s1}+T_{s2}+(m-k+1)T_2-(m-k+2+\delta)T_i\right\}\right]x$$
$$=f(s)-\left\{\delta T_{s1}+T_{s2}+mT_2-(m+1+\delta)T_i\right\}sf(s) \quad (31)$$

where it is understood that $a_m=1$, $a_{m+1}=0$, $a_{-1}=0$.

Since a feedback resistor in Fig. 4 will be connected to the first or second summer depending on the sign of a_k and on the order of the equation, the following general rule will apply:

$$a_k>0\begin{cases}a_{k\delta}=a_k \text{ if } (k+m) \text{ is odd}\\ =0 \text{ if } (k+m) \text{ is even.}\end{cases} \quad (32)$$
$$a_k<0\begin{cases}a_{k\delta}=0 \text{ if } (k+m) \text{ is odd}\\ =a_k \text{ if } (k+m) \text{ is even.}\end{cases} \quad (33)$$

The coefficients of (31) can be computed and compared with those of the given equation. By this means one can determine the effect of computer errors on the given equation even if the solution is not known. It would also be possible to perturb the coefficient-setting resistors or potentiometers by an amount equal to the error in coefficients given in (31) and to observe the effect of these perturbations on the computer solution, thus determining the sensitivity of the solution to these errors.

Solution of Simultaneous Differential Equations

The given set of equations in matrix notation is

$$[\dot{x}_i]=[a_{jk}][x_k]+[f_j] \quad (34)$$
$$\dot{X}=AX+F. \quad (35)$$

The computer setup to solve (34) is shown in Fig. 5, where it is assumed that all the coefficients a_{jk} are positive.

Using (13) and (20) it is possible, as before, to write the exact equation being solved by the machine. (Note that if any of the a_{jk} is negative, the corresponding potentiometer would be connected to the input of the integrator instead of to that of the summer. The symbol δ is used as before; $\delta=1$ when the input is to the summer; $\delta=0$ when it is connected to the integrator.) Then the voltage x_j is given by

$$[x_j]=\left[\frac{T_{ij}a_{jk}(T_is+1)^{1+\delta}}{(1+\delta T_{sj}s)(1+T_{1j}s)(1+T_2s)}\right][x_k]+\left[\frac{T_{1j}f_j(T_js+1)^{1+\delta}}{(1+\delta T_{sj}s)(1+T_{1j}s)(1+T_2s)}\right] \quad (36)$$

where

$T_{1j}=T_1$ for the jth integrator,
$T_{sj}=T_s$ for the jth summer,
$T_2=T_2$ for each integrator,
$T_i=R_iC_i$ for each summer and integrator.

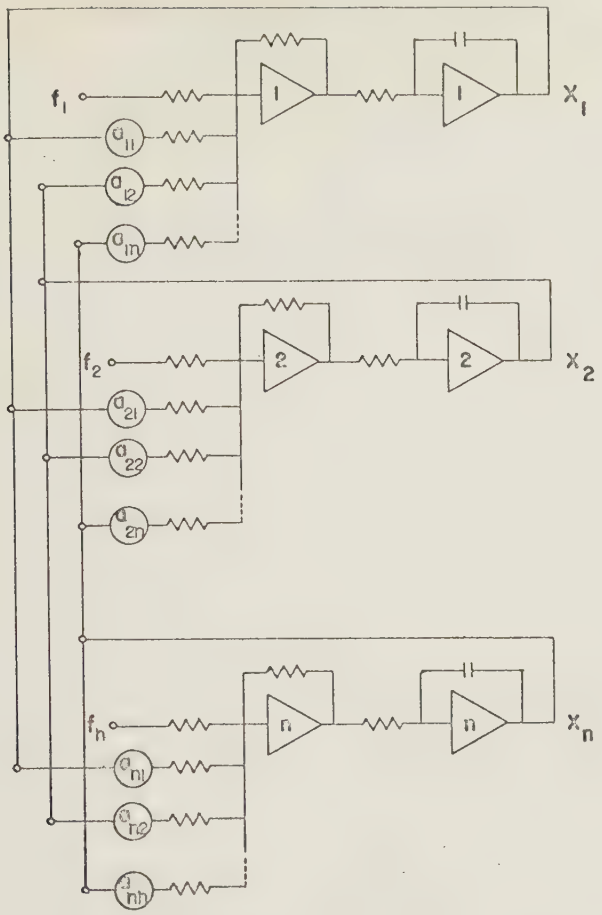


Fig. 5—Computer setup to solve simultaneous equations.

Eq. (36) is the exact equation solved by the computer and is seen to be at most of order $3n$ if all n summers are used. That is, it is of order $2n$ plus the number of summers.

Using approximations (29), (36) can be written

$$\left[a_{jk}\{1 - \delta T_{sj}s - T_2s + (1+\delta)T_is\} - I_{jk}\left(s + \frac{1}{T_{1j}}\right)\right][x_k] + [f_j\{1 - \delta T_{sj}s - T_2s + (1+\delta)T_1s\}] = 0 \quad (37)$$

where

$$I_{jk} = 1, \quad \text{if } j = k$$
$$= 0, \quad \text{if } j \neq k.$$

Finally (37) becomes

$$[a_{jk}\{\delta T_{sj} + T_2 - (1+\delta)T_i\} + I_{jk}][\dot{x}_k] = \left[a_{jk} - \frac{I_{jk}}{T_{1j}}\right][x_k] + [f_j - \{\delta T_{sj} + T_2 - (1+\delta)T_i\}\dot{f}_j]. \quad (38)$$

Let

$$B = [a_{jk}\{\delta T_{sj} + T_2 - (1+\delta)T_i\} + I_{jk}]$$
$$C = \left[a_{jk} - \frac{I_{jk}}{T_{1j}}\right]$$
$$G = [f_j - \{\delta T_{sj} + T_2 - (1+\delta)T_i\}\dot{f}_j]. \quad (39)$$

Then (38) can be written

$$B\dot{X} = CX + G. \quad (40)$$

Eq. (40) is a set of n first order simultaneous equations, as was the given equation. To facilitate comparison of the two equations, both sides of (40) are premultiplied by the inverse of B. Then

$$\dot{X} = B^{-1}CX + B^{-1}G. \quad (41)$$

The matrix B as given by (39) can be written

$$B = I + \Delta \quad (42)$$

where I is the identity matrix and the elements of the matrix Δ are small compared to unity. The inverse of B is given by

$$B^{-1} = I - \Delta + \Delta^2 - \cdots \quad (43)$$

and if the terms involving powers of Δ higher than the first are neglected in (43), one has

$$B^{-1} = [I_{jk} - a_{jk}\{\delta T_{sj} + T_2 - (1+\delta)T_i\}]. \quad (44)$$

Substituting (39) and (44) in (41) and neglecting products of small terms yields

$$[\dot{x}_j] = \left[a_{jk} - \frac{I_{jk}}{T_{1j}} - a_{jm}a_{mk}\{\delta T_{sj} + T_2 - (1+\delta)T_i\}\right][x_k] + [f_j - a_{jm}\{\delta T_{sj} + T_2 - (1+\delta)T_i\}f_m - \{\delta T_{sj} + T_2 - (1+\delta)T_1\}\dot{f}_j] \quad (45)$$

where the subscript m appearing in each term of a product denotes summation over m.

Eq. (45) is identical in form to (34) and the two can be compared term by term to determine the changes in coefficients resulting from computer errors.

Although only two computer setups have been considered here, the method of approach can be applied to any setup used to solve linear differential equations to determine the errors in the constant coefficients.

Example

As an example of the application of the method, consider the equation of simple harmonic motion

$$\ddot{x} + \omega^2 x = 0. \quad (46)$$

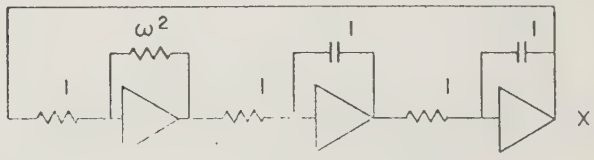


Fig. 6—Computer setup for simple harmonic motion.

Fig. 6 shows the computer setup, which is like the one shown in Fig. 4. The machine equation can be found from (31), and is

$$\left[s^2 + \left\{\frac{2}{T_1} - \omega^2(T_s + 2T_2 - 3T_i)\right\}s + \omega^2\right]x = 0. \quad (47)$$

Eq. (47) has a damping ratio given by

$$\zeta = \frac{1}{\omega T_1} - \frac{\omega(T_s + 2T_2 - 3T_i)}{2} \tag{48}$$

where T_s, T_i, T_1, and T_2 are defined by (14), (15), and (21).

Substituting (49) in (5) gives, for a summer,

$$V_0 = \frac{-\sum_i \frac{R_f}{R_i} V_i}{1 + \frac{(T_A s + 1)^2}{G_0(T_B s + 1)}\left(1 + \sum_i \frac{R_f}{R_i}\right)} \tag{50}$$

where the stray capacitance and resistance to ground have been ignored since they do not influence the result. Eq. (50) is rewritten as

$$V_0 = \frac{-G_0(T_B s + 1)\sum_i \frac{R_f}{R_i} V_i}{\left(1 + \sum_i \frac{R_f}{R_i}\right) T_A^2 s^2 + \left[2T_A\left(1 + \sum_i \frac{R_f}{R_i}\right) + G_0 T_B\right] s + \left[G_0 + 1 + \sum_i \frac{R_f}{R_i}\right]}. \tag{51}$$

Eq. (51) can be written

$$V_0 = \frac{-\sum_i \frac{R_f}{R_i} V_i}{1 + \frac{1}{G_0}\left(1 + \sum_i \frac{R_f}{R_i}\right) + \frac{2T_A - T_B}{G_0}\left(1 + \sum_i \frac{R_f}{R_i}\right) s + \frac{(T_A - T_B)^2}{G_0}\left(1 + \sum_i \frac{R_f}{R_i}\right) s^2 + \cdots}. \tag{52}$$

Appendix

Since many amplifiers in use actually have more complicated gain characteristics than indicated in Fig. 1, it is important to provide for the analysis of computers using such amplifiers. Since all amplifiers must have high gain at low frequencies (for good static accuracy) and must fall off at approximately −6 db per octave in the vicinity of unity gain (to insure stability when used as summers or integrators), Fig. 7 shows a frequency response characteristic which can fairly well be used to represent a wide variety of operational amplifiers.

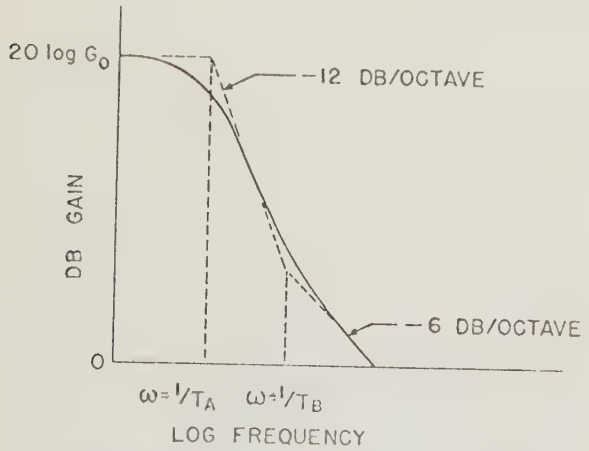


Fig. 7—Frequency response of open loop amplifier.

The transfer function for the amplifier of Fig. 7 is

$$G(s) = \frac{G_0(T_B s + 1)}{(T_A s + 1)^2}. \tag{49}$$

If

$$\left|\frac{1}{G_0}\left[1 + \sum_i \frac{R_f}{R_i}\right]\left[1 + (T_A - T_B)^2 s^2 + \cdots\right]\right| \ll 1,$$

(52) becomes

$$V_0 = \frac{-\sum_i \frac{R_f}{R_i} V_i}{1 + T_s s} \tag{53}$$

where

$$T_s = \frac{2T_A - T_B}{G_0}\left(1 + \sum_i \frac{R_f}{R_i}\right). \tag{54}$$

From (15) it is seen that this is equivalent to replacing the operational amplifier of Fig. 7 by one having the characteristics of Fig. 1, where

$$T_0 = 2T_A - T_B \qquad T_B \leqq T_A. \tag{55}$$

It can be shown by a similar procedure that (55) applies also when the operational amplifier is used as an integrator.

Acknowledgment

The author is grateful to R. M. Howe, the chairman of his doctoral committee, for his continued advice and counsel. He is indebted to the United States Air Force for providing him with the opportunity and facilities to pursue this investigation while associated with the Air Force Institute of Technology.

Synthesis of Vector Networks*

R. E. HORN† AND V. G. FAUQUE†

Summary—The convenience of vector notation in formulating physical geometrical problems results principally because the significance of the problem can be isolated from the analysis used in its solution. When analog computer techniques are employed in problems of this nature, the advantages of the vector methods frequently are lost because the computer inherently is more adaptable to solving problems described in the algebraic field of real numbers rather than in a vector space, and vector equations must be reduced to their scalar counterparts before a network for solving the equations may be synthesized. To facilitate treatment of problems of this type, a method is presented for synthesizing networks directly from the vector notation. This method will simplify synthesis and analysis of the networks by drawing a closer analogy between the mathematics and the electronics.

The application of transformations and operators represents two areas in which the pertinent aspects of the method can be illustrated. This is accomplished by presenting first the basic mathematical background, followed by the network aspects of the problem, and finally, by the illustrative examples. The more extensive problems usually encountered, for example, in airborne fire-control systems, then may be synthesized by proper application of the elementary "vector networks."

I. Transformations

CONSIDER two sets of three-dimensional Cartesian coordinates. Let X_1, X_2, X_3, and Y_1, Y_2, Y_3 denote mutually orthogonal axes of the two systems. Bases for vectors in either system will be orthonormal and colinear with the coordinate axes. These bases shall be denoted by $\boldsymbol{i}_1$, $\boldsymbol{i}_2$, $\boldsymbol{i}_3$, and $\boldsymbol{j}_1$, $\boldsymbol{j}_2$, $\boldsymbol{j}_3$ in the X and Y systems, respectively. Let the $\boldsymbol{i}$ and $\boldsymbol{j}$ bases be related as follows:

$$\begin{aligned}\boldsymbol{i}_1 &= f_{11}\boldsymbol{j}_1 + f_{12}\boldsymbol{j}_2 + f_{13}\boldsymbol{j}_3,\\ \boldsymbol{i}_2 &= f_{21}\boldsymbol{j}_1 + f_{22}\boldsymbol{j}_2 + f_{23}\boldsymbol{j}_3,\\ \boldsymbol{i}_3 &= f_{31}\boldsymbol{j}_1 + f_{32}\boldsymbol{j}_2 + f_{33}\boldsymbol{j}_3.\end{aligned} \tag{1}$$

In matrix notation, this relation is expressed as

$$\begin{bmatrix}\boldsymbol{i}_1\\ \boldsymbol{i}_2\\ \boldsymbol{i}_3\end{bmatrix} = \begin{bmatrix}f_{11} & f_{12} & f_{13}\\ f_{21} & f_{22} & f_{23}\\ f_{31} & f_{32} & f_{33}\end{bmatrix}\begin{bmatrix}\boldsymbol{j}_1\\ \boldsymbol{j}_2\\ \boldsymbol{j}_3\end{bmatrix} = \|f_{ij}\|\begin{bmatrix}\boldsymbol{j}_1\\ \boldsymbol{j}_2\\ \boldsymbol{j}_3\end{bmatrix}. \tag{2}$$

Consider a vector $\boldsymbol{r}$ such that

$$\boldsymbol{r} = g_1\boldsymbol{i}_1 + g_2\boldsymbol{i}_2 + g_3\boldsymbol{i}_3, \tag{3}$$

and

$$\boldsymbol{r} = h_1\boldsymbol{j}_1 + h_2\boldsymbol{j}_2 + h_3\boldsymbol{j}_3. \tag{4}$$

Since a vector defined by a complete and independent set of bases has a unique set of components, the transformation relating any two sets of bases also relate their respective components. Therefore,

$$\begin{bmatrix}g_1\\ g_2\\ g_3\end{bmatrix} = \|f_{ij}\|\begin{bmatrix}h_1\\ h_2\\ h_3\end{bmatrix}. \tag{5}$$

Multiplying the left-hand side of (5) by the inverse transformation $\|f_{ij}\|^{-1}$ yields

$$\begin{bmatrix}h_1\\ h_2\\ h_3\end{bmatrix} = \|f_{ij}\|^{-1}\begin{bmatrix}g_1\\ g_2\\ g_3\end{bmatrix}. \tag{6}$$

Since the $\|f_{ij}\|$ transformation is orthogonal, its inverse is equal to its transpose. Therefore

$$f_{ij}\|^{-1} = \begin{bmatrix}f_{11} & f_{21} & f_{31}\\ f_{12} & f_{22} & f_{32}\\ f_{13} & f_{23} & f_{33}\end{bmatrix}, \tag{7}$$

and (6) may be written

$$\begin{bmatrix}h_1\\ h_2\\ h_3\end{bmatrix} = \begin{bmatrix}f_{11} & f_{21} & f_{31}\\ f_{12} & f_{22} & f_{32}\\ f_{13} & f_{23} & f_{33}\end{bmatrix}\begin{bmatrix}g_1\\ g_2\\ g_3\end{bmatrix}. \tag{8}$$

II. Operators

Suppose that the rate of change of the vector $\boldsymbol{r}$ is to be expressed in terms of its components on the $\boldsymbol{i}$ base system, which is rotating. That is,

$$d\boldsymbol{r} = m_1\boldsymbol{i}_1 + m_2\boldsymbol{i}_2 + m_3\boldsymbol{i}_3, \tag{9}$$

where d is the operator d/dt[1] and $\boldsymbol{r} = \boldsymbol{r}(t)$. With $\boldsymbol{r}$ defined by

$$\boldsymbol{r} = g_1\boldsymbol{i}_1 + g_2\boldsymbol{i}_2 + g_3\boldsymbol{i}_3, \tag{3}$$

the derivative of $\boldsymbol{r}$ is by definition

$$d\boldsymbol{r} = (dg_1)\boldsymbol{i}_1 + (dg_2)\boldsymbol{i}_2 + (dg_3)\boldsymbol{i}_3 + \boldsymbol{\omega} \times \boldsymbol{r}, \tag{10}$$

where

$$\boldsymbol{\omega} = \omega_1\boldsymbol{i}_1 + \omega_2\boldsymbol{i}_2 + \omega_3\boldsymbol{i}_3 \tag{11}$$

defines the angular velocity of the X-coordinate system with reference to some fixed system. Using conventional vector methods,

$$d\boldsymbol{r} = (dg_1)\boldsymbol{i}_1 + (dg_2)\boldsymbol{i}_2 + (dg_3)\boldsymbol{i}_3 + \begin{vmatrix}\boldsymbol{i}_1 & \boldsymbol{i}_2 & \boldsymbol{i}_3\\ \omega_1 & \omega_2 & \omega_3\\ g_1 & g_2 & g_3\end{vmatrix}, \tag{12}$$

* Manuscript received by the PGEC, June 19, 1957; revised manuscript received, October 1, 1957.

† Westinghouse Electric Corp., Baltimore, Md.

[1] The operator d is not to be confused with a multiplier. The derivative operation is not distributive with respect to multiplication.

which may be alternately expressed as

$$dr = (dg_1 - \omega_3 g_2 + \omega_2 g_3)i_1 + (\omega_3 g_1 + dg_2 - \omega_1 g_3)i_2 + (-\omega_2 g_1 + \omega_1 g_2 + dg_3)i_3. \quad (13)$$

From (9) and (13) the equivalence

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} dg_1 - \omega_3 g_2 + \omega_2 g_3 \\ \omega_3 g_1 + dg_2 - \omega_1 g_3 \\ -\omega_2 g_1 + \omega_1 g_2 + dg_3 \end{bmatrix}, \quad (14)$$

or the factored form

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} d & -\omega_3 & \omega_2 \\ \omega_3 & d & -\omega_1 \\ -\omega_2 & \omega_1 & d \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} \quad (15)$$

may be recognized. It should be noted that (15) also may be derived directly using matrix methods.[2]

The matrix

$$\begin{bmatrix} d & -\omega_3 & \omega_2 \\ \omega_3 & d & -\omega_1 \\ -\omega_2 & \omega_1 & d \end{bmatrix} = \| D\omega \| \quad (16)$$

shall be designated by the term "operator matrix." The product of two operator matrices will provide a second-order operator matrix. In general, $\| D\omega \|^n$ will be an nth order operator matrix, where the superscript indicates the number of multiplication of the operator matrix by itself.

For use in the synthesis of networks, Section IV shows that the matrices need not be combined before synthesis.

To differentiate a vector in a fixed coordinate system, the operator matrix (16) may be used, but since the ω's are zero, the operator matrix becomes

$$\begin{bmatrix} d & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & d \end{bmatrix} = \| D \|. \quad (17)$$

The integration matrix is defined as the inverse of the operator matrix. Similarly, the expression

$$\int \boldsymbol{r} dt \equiv d^{-1}\boldsymbol{r} \quad (18)$$

is defined. Now let

$$d^{-1}\boldsymbol{r} = q_1 \boldsymbol{i}_1 + q_2 \boldsymbol{i}_2 + q_3 \boldsymbol{i}_3. \quad (19)$$

Then

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \| D\omega \|^{-1} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} \quad (20)$$

or, since the matrix $\| D\omega \|$ always possesses a right inverse,

$$\| D\omega \| \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}, \quad (21)$$

which may be solved for $\| q \|$ when the initial values of $\| q \|$ are known.

III. Networks

Synthesis of equations of the general form

$$\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} e_{11}' & e_{12}' & e_{13}' \\ e_{21}' & e_{22}' & e_{23}' \\ e_{31}' & e_{32}' & e_{33}' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \quad (22)$$

will be considered, since a correspondence exists between (22) and any equations discussed previously. The network which approximates[3] the analog of (22) will be general in the sense that any illustrative example will be a special case of this configuration. The basis for the synthesis is the one-to-one correspondence between (22) and idealized admittance equation. This idealized admittance equation may be written as

$$\begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix} \begin{bmatrix} -w_1 \\ -w_2 \\ -w_3 \end{bmatrix} = \begin{bmatrix} y_{11}' & y_{12}' & y_{13}' \\ y_{21}' & y_{22}' & y_{23}' \\ y_{31}' & y_{32}' & y_{33}' \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \quad (23)$$

where the coefficient matrices $\| y \|$ and $\| y' \|$ represent network admittances, the u's the independent voltage variables, and the w's the dependent voltage variables.

Treatment of network equations of this type, as applied to analog computers, has been covered elsewhere,[4,5] and therefore, it should suffice to discuss briefly the schematic representation to be used.

The basic network consisting of an amplifier with gain u_k having bipolarity inputs, and associated admittances Y_{jk} is shown in Fig. 1. Input grid voltages are represented by u's and output voltages by $+w_k$ and $-w_k$. By representing the network in this way rather than as a conventional computer schematic, recognition of the basic matrix form is simplified.

Fig. 2 gives the final representation, as its combination with additional networks of the same type. The principal changes have been to draw the amplifier as numbered squares and to eliminate the representation of the common reference mode. The schematic for the

[2] R. A. Frazer, W. J. Duncan, and A. R. Collar, "Elementary Matrices," Cambridge Univ. Press, Cambridge, Eng., pp. 252–253; 1938.

[3] Networks which are exactly analogous to the equation are not realizable since the gain of the amplifiers would necessarily be nonfinite. A detailed description of this problem is given by Honnell and Horn, footnote reference 5.

[4] P. M. Honnell and R. E. Horn, "Matrices in analogue mathematical machines," *J. Franklin Inst.*, vol. 260, pp. 193–207; September, 1955.

[5] P. M. Honnell and R. E. Horn, "Analogue computer synthesis and error matrices," *AIEE Trans., Part I—Commun. and Electronics*, vol. 75, pp. 26–32; March, 1956.

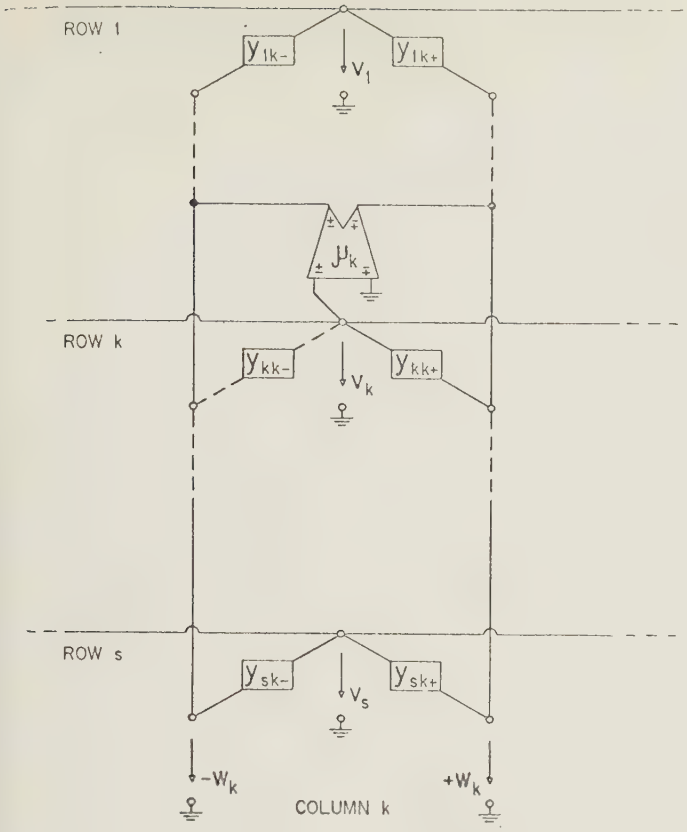


Fig. 1—Basic computing network.

prescribed voltage sources $+u_k$ and $-u_k$ and corresponding admittances Y_{jk}, although not drawn in Fig. 2, is represented by similar grid structure. By arranging the schematic as a rectangular grid, the direct correspondence between the matrically-formulated problem and the computer network is emphasized.

The general form of the equations considered in this paper is given as (23). Using the notation described in the preceding paragraphs, the synthesis of (23) is illustrated in Fig. 3. Required amplifiers are shown as numbered squares, with horizontal lines (v's) representing input grids. The vertical lines indicate measured variables, those in the left section of the array representing bipolarity amplifier outputs (w's), those in the right section representing prescribed voltage sources (u's). Admittance elements are connected in positions corresponding to terms of the problem matrix and, although only positive matrix terms are illustrated in Fig. 3, negative signs are obtained readily by utilizing the negative voltage variables. With the polarity convention indicated, principal diagonal terms on the left side of the network equation (23) normally must be positive for network stability.

To synthesize networks corresponding to the transformation and operator matrices already described, it is only necessary to relate each term in the coefficient matrices to the corresponding network parameter in the computer. This will be illustrated by specific examples in Section IV.

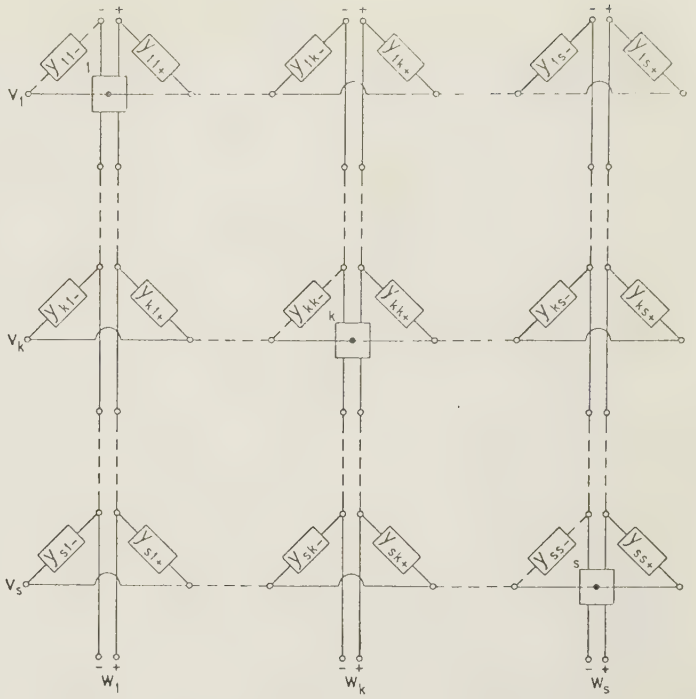


Fig. 2—Schematic representation of combined networks.

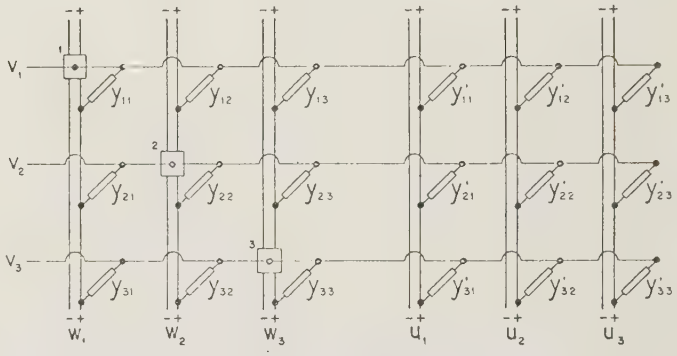


Fig. 3—Network corresponding to (22) and (23).

IV. Examples

A. Transformations

Transformations through Eulerian angles are the common types requiring synthesis. For example, rotation of the X system about the Y_1 axis through an angle of ϕ (in a right-handed sense) requires the $||f_{ij}||$ transformation of Section I to be

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}, \tag{24}$$

so that (5) becomes

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}. \tag{25}$$

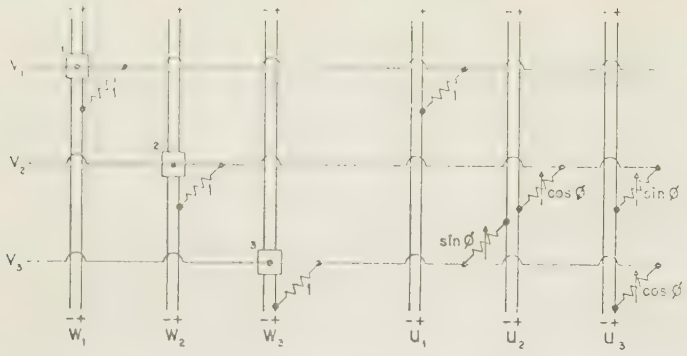


Fig. 4 - Network corresponding to (25) and (26).

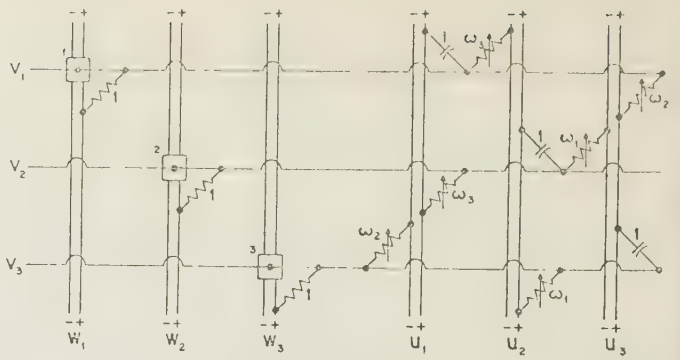


Fig. 5 - Network corresponding to (15) and (28).

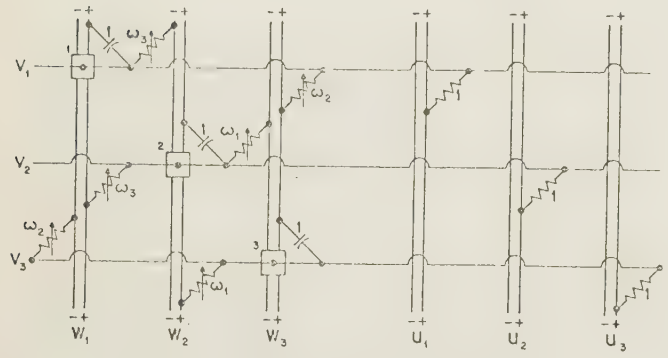


Fig. 6—Network corresponding to (21).

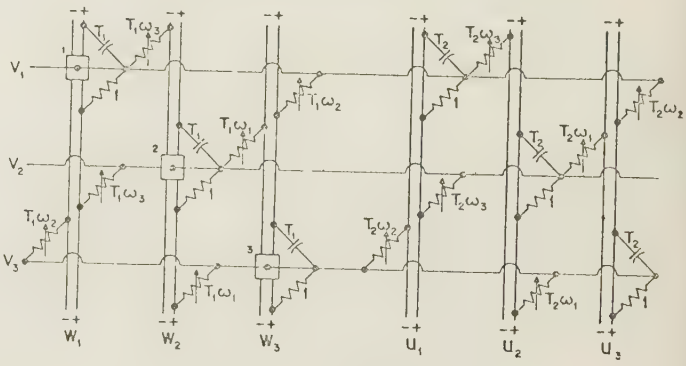


Fig. 7—Network corresponding to (32).

The corresponding network equation[6]

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -w_1 \\ -w_2 \\ -w_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{26}$$

then may be synthesized directly as indicated in Fig. 4. Note that the formal writing of (26) is unnecessary since the network of Fig. 4 is obtainable directly from (25). The varying admittances required for the variable ϕ are conventionally obtained by the use of servodriven potentiometers and series conductances which yield the required transfer admittance.

Frequently, transformations involving three Euler angles are required. These take the form

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$\cdot \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}. \tag{27}$$

[6] The matrix $\begin{bmatrix} 100 \\ 010 \\ 001 \end{bmatrix}$ is the identity matrix. It is used here so that this example will conform to the general form described in (22).

In this case, three networks of the type shown in Fig. 4 are connected in cascade. Of course, this is equivalent to multiplying the matrix by the $\|\psi\|$ matrix, the product by $\|\theta\|$, and that product by the $\|\phi\|$ matrix. Since the form of each network has already been established, it is unnecessary to perform the multiplication indicated in (27), but only to synthesize the three networks with terms as required. This process not only simplifies the initial mathematical problem, but it provides a simpler synthesis and analysis of the network.

The inverse transformations, as discussed in Section I, have a similar form to the transformations themselves. Therefore, the synthesis of networks involving inverse orthogonal transformations reduces to the identical process as that explained above. This avoids the question of stability which necessarily arises for networks involving "cross coupling" between amplifiers.

B. Differentiation

Differentiation of a vector, described by

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} d & -\omega_3 & \omega_2 \\ \omega_3 & d & -\omega_1 \\ -\omega_2 & \omega_1 & d \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}, \tag{15}$$

and the computer equation

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -w_1 \\ -w_2 \\ -w_3 \end{bmatrix} = \begin{bmatrix} d & -\omega_3 & \omega_2 \\ \omega_3 & d & -\omega_1 \\ -\omega_2 & \omega_1 & d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \tag{28}$$

may be synthesized as illustrated in Fig. 5. The one-to-one correspondence between terms in (15) and network elements is evident again.

C. Integration

Integration of the r vector in the X-coordinate system, as discussed in Section II may be expressed as

$$\begin{bmatrix} d & -\omega_3 & \omega_2 \\ \omega_3 & d & -\omega_1 \\ -\omega_2 & \omega_1 & d \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}, \tag{21}$$

representing an inverse of the operation on the $\|g\|$ matrix in (15). The network corresponding to (21) is shown in Fig. 6.

D. Filtering

The problem of filtering "noise" in a moving coordinate system is illustrative of a simple practical application. As previously defined, let

$$r = g_1 i_1 + g_2 i_2 + g_3 i_3 \tag{3}$$

represent the vector input to the filter and

$$p = p_1 i_1 + p_2 i_2 + p_3 i_3 \tag{29}$$

represent the output vector. Then the vector equation

$$(1 + T_1 d)p = (1 + T_2 d)r \tag{30}$$

is representative of the vector "lead-lag" network to be synthesized. In matrix notation, (30) may be written

$$\{\|I\| + T_1\|D_\omega\|\}\|p\| = \{\|I\| + T_2\|D_\omega\|\}\|g\| \tag{31}$$

or

$$\begin{bmatrix} (1 + T_1 d) & -T_1\omega_3 & T_1\omega_2 \\ T_1\omega_3 & (1 + T_1 d) & -T_1\omega_1 \\ -T_1\omega_2 & T_1\omega_1 & (1 + T_1 d) \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} (1 + T_2 d) & -T_2\omega_3 & T_2\omega_2 \\ T_2\omega_3 & (1 + T_2 d) & -T_2\omega_1 \\ -T_2\omega_2 & T_2\omega_1 & (1 + T_2 d) \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} \tag{32}$$

with the corresponding network illustrated in Fig. 7.

In (32), an insight into the stability question may be obtained if the constant coefficient case is assumed, resulting in a determinantal equation having roots

$$\lambda_1 = -\frac{1}{T_1}, \tag{33}$$

$$\lambda_{2,3} = -\frac{1}{T_1} \pm j\sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2}. \tag{34}$$

Eq. (34) indicates the cross-coupling effects introduced by the rotating coordinate system.

V. Conclusion

By indicating the direct relationship between commonly-used vector equations and matrix synthesis methods, a convenience is achieved that may be lost by conventional methods. Specific examples illustrating transformation and derivative matrices are representative of the type problems that arise, and the corresponding basic configurations serve as fundamental "vector networks" to be used as units in the synthesis of more complex problems.

Switching Functions of Three Variables*

D. W. DAVIES†

Summary—A switching function is a function of variables which take only the values 0 and 1, and which takes only these values itself. There are 256 different switching functions of three variables, but only 218 of these really depend on all three variables.

A switching function of three variables can be expressed in terms of switching functions of two variables. For example

$$F_1\{F_2[A, F_3(B, C)], F_4(B, C)\}$$

can be shown to represent any function of A, B, and C if $F_1F_2F_3$ and F_4 are suitably chosen switching functions.

The problem solved is: for each switching function of three variables, what is the least number of switching functions of two variables required to express it?

This problem leads to a discussion of the symmetry of switching functions. Expressions which, like the one above, can represent any switching function of three variables are investigated. Expressions which, by permutation of variables, can represent any switching function of three variables are also determined.

The investigation is done by means of "logical diagrams," which give a better intuitive understanding than the functional expressions.

IN THE logical design of digital computers, electrical signals which have only two significant levels are often represented by variables which can take only two values 0 and 1. Let A, B, and C be three such variables. Suppose now that the three signals represented by A, B, and C enter an unspecified unit and this unit produces an output which depends only on the inputs and which also has only two significant levels,

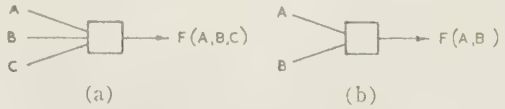
* Manuscript received by the PGEC, July 17, 1957.

† National Physical Lab., Teddington, Middlesex, Eng. The work in connection with this paper has been carried out as a part of the research program of the National Physical Laboratory, and the paper is published by permission of the Director of the Laboratory.

represented by 0 and 1. Such an arrangement is shown in Fig. 1(a). The output signal is expressed as

$$F(A, B, C)$$

which is called a switching function of the three variables A, B, and C.



(a) (b)

Fig. 1 Nomenclature for "black boxes."

An unspecified unit such as we have described is often called a "black box." We are reserving the term here for a unit with no memory; *i.e.*, one in which the output depends only on the present values of the inputs.

It is well known that a switching function of n variables can be synthesized by using an arrangement of several black boxes each with only two inputs, like that of Fig. 1(b).

The question answered here is: for each switching function of three variables, what is the minimum number of black boxes (of any type) with two inputs which can be connected together to synthesize it? A related question is: given a circuit arrangement of black boxes with two inputs, the whole having three inputs and an output but nothing being known about the boxes themselves, what is the class of switching functions the given function of three variables is known to belong to? This is answered for all cases of four boxes or less.

Since black boxes with two inputs are important to our problem we shall adopt the convention that "black box" means one of these.

The problem could be stated more accurately thus: for each switching function of three variables, what is the least number of switching functions of two variables required to express it? As an example, take the circuit arrangement of Fig. 17(a). This can be expressed as

$$F(A, B, C) = F_1\{F_2(A, C), F_3(C, B)\}.$$

The three boxes correspond to the three functions F_1, F_2, and F_3. In this paper we use the logical diagrams to represent this type of equation.

Functions of 0, 1, and 2 Variables

The following remarks concerning functions of less than three variables are perhaps obvious to anyone conversant with Boolean algebra or its related subjects, but they are given because they can be developed later for three variables.

There are two functions of 0 variables, 0, and 1.

A single variable can take two different values. If output, 0 or 1 is specified for each input we can have four functions of a single variable. These would be

$$F(A) = 0, 1, A, \sim A,$$

where $\sim A$ is the negation of A.

A pair of variables can take four different sets of values. If the output, 0 or 1 is specified for each input, we have $2^4 = 16$ functions of two variables. Table I is a list of the variables of $F(A, B)$, giving the four values it takes for the cases $(A, B) = (0, 0)$, $(0, 1)$, $(1, 0)$, and $(1, 1)$. The functions have been numbered 0–15, a notation which will be used later.

TABLE I

Nomenclature for Switching Functions of Two Variables

A	B	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

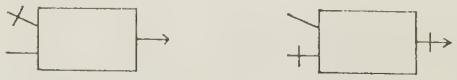
It is obvious some cases listed as $F(A, B)$ are really functions of just A or B, or of neither. Functions numbered 0 and 15 are functions of no variables, and functions 3, 5, 10, and 12 are $\sim A$, $\sim B$, B, and A, respectively. These are called "degenerate." There are 10 nondegenerate functions of two variables, two nondegenerate functions of one variable, and two of no variables.

The particular black box A $F(A)$ which gives $F(A) = \sim A$, will be represented $\twoheadrightarrow$ in what follows. The only other nondegenerate function of one variable if $F(A) = A$, which does not need a symbol.

If $F(A, B)$ is a general function of two variables, then so are

	$\sim F(A, B)$
$F(\sim A, B)$	$\sim F(\sim A, B)$
$F(A, \sim B)$	$\sim F(A, \sim B)$
$F(\sim A, \sim B)$	$\sim F(\sim A, \sim B)$.

Therefore, symbols such as



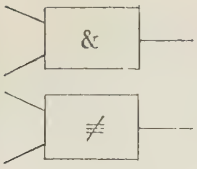
if the function is unspecified, can all be represented as

in which the negation has been absorbed into the black box. Suppose now, that a supply of negators is available how many black boxes with two inputs are needed to construct any function?

It is easily seen that functions 1, 2, 4, 7, 8, 11, 13, and 14 are related by negating A or B or F or interchanging A and B.

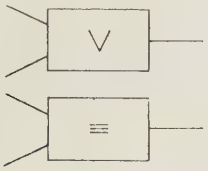
The remaining nondegenerate functions, 6 and 9, are related to one another.

By making use of the $\twoheadrightarrow$ symbol in what follows, we need employ only two different functions of two variables, one representative of each class. We choose symmetric functions which we symbolize as follows:



represents function 8,
(intuitively A and B).
represents function 6,
(intuitively A not equivalent to B).

It is convenient, however, to have symbols for two other functions which appeal to intuition namely



representing function 14,
(intuitively A or B).
representing function 9,
(intuitively A equivalent to B).

Representation of Functions of Three Variables

A set of three variables can take eight different values. The function is specified when its value is given for each of the eight values of the inputs. There are, therefore, $2^8 = 256$ different functions. This includes the following degenerate cases:

0,1

A, $\sim A$, B, $\sim B$, C, $\sim C$,

all the nondegenerate functions $F(A, B)$, $F(B, C)$, $F(A, C)$ of two variables.

These are 38 in numbers, leaving 218 nondegenerate functions of three variables.

The set of points (A, B, C) in Cartesian coordinates given by A, B, and C taking the values 0 and 1 is the corners of a cube. A representation of $F(A, B, C)$ is given by attaching the value of F to each corner. For example, Fig. 2 is a symmetric function of A, B, and C, that is to say, for this function $F(A, B, C) = F(B, A, C) = F(C, B, A) = F(C, A, B) = F(C, A, B) = F(B, C, A) = F(A, C, B)$. For this particular function, the representation on a cube exhibits a further kind of symmetry,

$$F(\sim A, \sim B, \sim C) = \sim F(A, B, C)$$

because reflection in the center of the cube (inversion) gives $\sim F$.

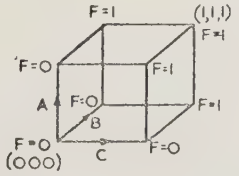


Fig. 2—Representation of $F(A, B, C)$ on a cube.

The representation as a labelled cube, therefore, shows up symmetries in the function. The operations on the labelled cube which map it on itself and can give rise to symmetry are:

$$A \rightarrow \sim A$$

$$B \rightarrow \sim B$$

$$C \rightarrow \sim C$$

$$F \rightarrow \sim F$$

$$B \leftrightarrows A, \text{ etc.}$$

These form a group G of order 96. It is the direct product of four cyclic groups of order 2 and the group of permutations of A, B, and C of order 6.

Any form of symmetry of the marked cube is given by a subgroup of G under all of whose operations $F(A, B, C)$ is invariant.

Suppose now that we have considered a particular $F(A, B, C)$ and know how to produce it by connecting together some two input black boxes. Interchange of inputs is a trivial operation and so is negation of inputs and outputs since any such negations can be absorbed into the black boxes to which they are connected. Acting in these ways from the original $F(A, B, C)$, in principle, 96 different functions such as, for example,

$$\sim F(B, \sim C, A)$$

can be formed. Actually, because of symmetry, some of these might be the same. If the order of the symmetry subgroup is n, there will in fact be $96/n$ different functions. This reduced set of functions will be called a "class" of functions. When one member of a class of functions has been realized by some arrangement of black boxes, all the other members of the class can be realized by trivial alterations to the arrangement.

The synthesis of switching functions of three variables has been treated,[1] but a restricted set of black boxes was employed to correspond with a certain set of electronic circuits. The symmetry classes were enumerated, but there were more classes than we find here, because the symmetry operation

$$F \rightarrow \sim F$$

(negation of output) was not employed. Slepian[2] gives a method for calculating the number of symmetry classes for switching functions of n variables but here also only input transformations were employed and the output transformation $F \rightarrow \sim F$ was not included.

A second method of representing functions of three variables will be used in this paper in order to show all 256 functions at once. If the variable C is held fixed, a function of A and B is obtained. There are two such functions

$$F(A, B, 0) = \phi_0(A, B)$$

$$\text{and } F(A, B, 1) = \phi_1(A, B),$$

which together determine completely the function $F(A, B, C)$. Each of the functions ϕ can have sixteen different values. Thus all the 256 functions can be represented in a 16×16 array. As an example, we give in Fig. 3 a picture of all the degenerate functions, and all those

[1] Harvard Computation Lab. Staff, "Synthesis of Electronic Computing and Control Circuits," Harvard Univ. Press, Cambridge, Mass., ch. 3; 1951.

[2] D. Slepian, "On the number of symmetry types of Boolean functions of n variables," *Can. J. Math.*, vol. 5, pp. 185–193; 1953.

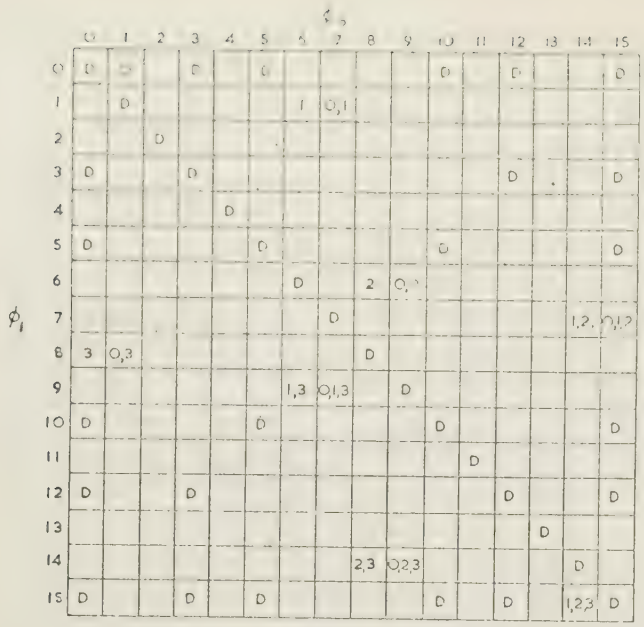


Fig. 3—The degenerate functions and symmetric functions. Degenerate functions are shown by a D: symmetric functions are shown by the values of $A+B+C$ which make $F=1$. The functions (0, 0) and (15, 15) are both degenerate and symmetric.

which are symmetric under the permutations of (A, B, C).

The numbering of the coordinate axes in Fig. 3, which represent values of ϕ_0 and ϕ_1 corresponds to the numbering of functions in Table I.

Functions Which Can Be Realized by Two Black Boxes

Two "black boxes," each with two inputs, can be connected only as in Fig. 4 to make a three-input "black box."

Fig. 4—The only arrangement of two "black boxes."

To evaluate all the function realizable by this array, all permutations of A, B, and C among the three inputs must be considered. The easiest arrangement, bearing in mind Fig. 3, is to connect C to input 3. We shall consider this first.

Suppose that the black box x gives an output $G(A, B)$. Then setting $C=0$, the output is a function of $G(A, B)$. This can be 0, 1, $G(A, B)$ or $\sim G(A, B)$.

Hence

$$\phi_0 = 0,\ 1,\ G \text{ or } \sim G,$$

similarly

$$\phi_1 = 0,\ 1,\ G \text{ or } \sim G.$$

The function G can be any of the 16 functions of two variables. There are six cases:

$$\phi_0 = 0 \qquad \phi_1 = 0, 1, 2, \cdots, \text{or } 15,$$
$$\phi_0 = 1 \qquad \phi_1 = 0, 1, 2, \cdots, \text{or } 15,$$
$$\phi_1 = 0 \qquad \phi_0 = 0, 1, 2, \cdots, \text{or } 15,$$
$$\phi_1 = 1 \qquad \phi_0 = 0, 1, 2, \cdots, \text{or } 15,$$
$$\phi_0 = \phi_1 = 0, 1, 2, \cdots, \text{or } 15,$$
$$\phi_0 = \sim \phi_1 = 0, 1, 2, \cdots, \text{or } 15,$$

These give 88 functions lying on both diagonals and around the border of the 16×16 chart.

Next consider the case of B applied to the input 3. Suppose that the box y gives an output $G(x, B)$ where x is the output of black box x.

If $C=0$ we have

$$\phi_0 = G(0, B),\ G(1, B),\ G(A, B), \text{ or } G(\sim A, B).$$

If $C=1$, also

$$\phi_1 = G(0, B),\ G(1, B),\ G(A, B), \text{ or } G(\sim A, B).$$

The only cases of interest are:

$$\phi_0 = G(0, B) \qquad \phi_1 = G(A, B),$$
$$\phi_0 = G(1, B) \qquad \phi_1 = G(A, B),$$
$$\phi_0 = G(A, B) \qquad \phi_1 = G(A, B)$$

which is already covered, and

$$\phi_0 = G(A, B) \qquad \phi_1 = G(\sim A, B),$$

and those obtained by interchanging ϕ_0 and ϕ_1.

It only remains to consider the case of A applied to input 3. This is the same as the last case with A and B interchanged. Interchange of A and B is a transformation of the ϕ's and is easily applied.

If all these possibilities are enumerated and plotted on the 16×16 chart, a total of 152 functions are found to be realizable by means of two black boxes. These include all the 38 degenerate cases.

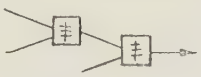
Obviously, the 114 nondegenerate cases can be divided into symmetry classes because if any function is realizable in two black boxes so are its transforms under the symmetry operators. We list in Table II the five symmetry classes, the order of their symmetry group and the number of different functions in the class. A typical member of the class, one exhibiting obvious symmetry, is shown also.

The use of the negator symbol is unnecessary, but it enables the black boxes to be restricted to & and $\not\equiv$.

It is known that & and $\not\equiv$ boxes, with negators are sufficient to produce any two-input box. Negators on the inputs and outputs of the boxes in Fig. 4 produce functions of the same class, so the five arrangements shown above are the only possible ones. Negators on the connecting link between the boxes in cases a, c, and d can be transferred via a $\not\equiv$ box to the inputs or outputs, where they are unnecessary. Table II, therefore, gives another proof that only 114 nondegenerate functions are realizable by two boxes.

TABLE II

THE SYMMETRY CLASSES REALIZABLE BY TWO "BLACK BOXES"

Symbol Used for Class	Typical Member	Order of Symmetry Group	No. in Class
a		48	2
b		6	16
c		4	24
d		4	24
e		2	48
		Total	114

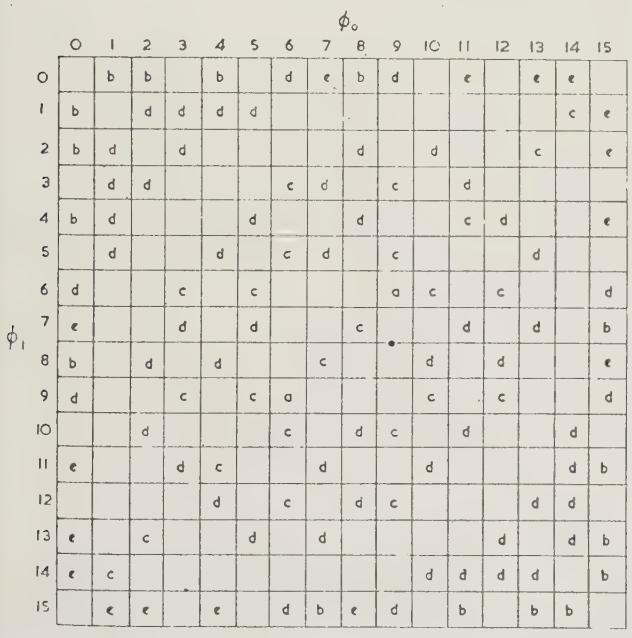


Fig. 5—The symmetry classes *a*, *b*, *c*, *d*, and *e*.

These functions are plotted in Fig. 5 on the 16×16 chart, the class symbol of each function being shown.

FUNCTIONS WHICH CAN BE REALIZED BY THREE BLACK BOXES, BUT NOT TWO

Three black boxes can be connected in only two ways to produce new functions of three variables. These are shown in Figs. 6 and 7.

To enumerate all the possibilities arising from Fig. 6,

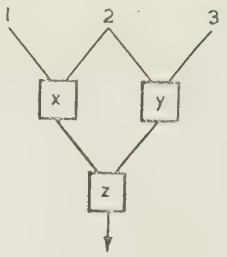


Fig. 6—The first arrangement of three "black boxes."

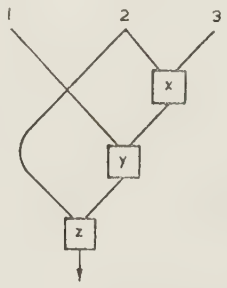


Fig. 7—The second arrangement of three "black boxes."

we start by considering the possible values of functions x and y. We are only interested initially in getting representatives from each symmetry class, so negations of inputs 1, 2, or 3 or of the output, and interchanges of inputs are irrelevant. Therefore, only the possibilities & and $\not\equiv$ for x and y, with a possible negation on one of the two branches from input 2, need be considered. The case of x and y both & reduces to the case of two boxes. The negation is irrelevant if either x or y is $\not\equiv$. Therefore, only three useful cases arise:

x	y
1 & 2	(~ 2) & 3
1 & 2	$2 \not\equiv 3$
$1 \not\equiv 2$	$2 \not\equiv 3$.

These must be considered with all nondegenerate functions Z, but regarding negation of the output as irrelevant. Functions 1, 2, 4, 6, and 8 can be used. Of the 15 possibilities, all but 8 are found to be covered by the two box representation.

These eight functions and all their transforms under symmetry operations were plotted on the 16×16 chart. It was found that only three were essentially different, that is there are three new classes generated by the arrangement of Fig. 6.

The new classes, representative members of the classes and the order of their symmetry group are listed in Table III. The classes are plotted on the 16×16 chart in Fig. 8.

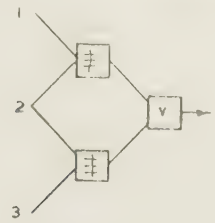
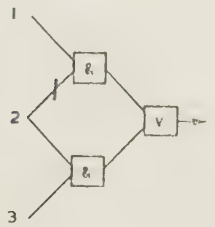
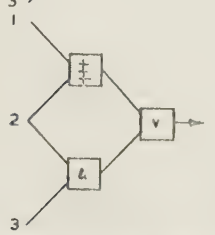
We still have to consider the arrangement shown in Fig. 7. This is relatively simple.

Suppose C, A, and B are connected to the 1, 2, and 3 inputs. Let the output of x be

$$G(A, B).$$

TABLE III

THE SYMMETRY CLASSES REALIZABLE BY THREE "BLACK BOXES"

Symbol Used for Class	Typical Member	Order of Symmetry Group	No. in Class
f		12	8
g		4	24
h		2	48
		Total	80

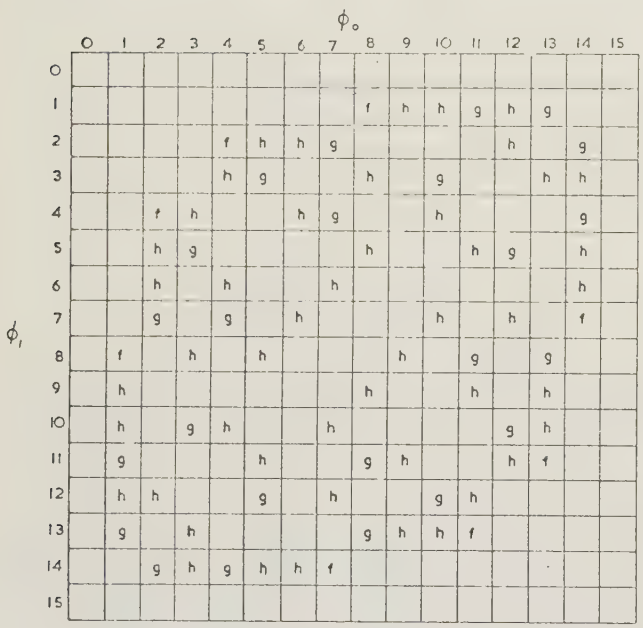


Fig. 8—The symmetry classes f, g, and h.

Then the values of the output of y for $C=0$ and $C=1$ are

$$y_0,\ y_1,\ = 0,\ 1,\ G \text{ or } \sim G.$$

For the box Z we need consider only the possibilities

$$Z = A\ \&\ y, \qquad Z = A \not\equiv y.$$

When all these are considered, the only cases not covered by the same arrangement as Fig. 7 with box Z removed are found to be the ones like

$$\phi_0 = A, \qquad \phi_1 = A\ \&\ G$$

and

$$\phi_0 = A, \qquad \phi_1 = A \not\equiv G.$$

The first of these gives only degenerate functions and functions realizable by two boxes.

The second gives some of the functions plotted in Fig. 8, those on the columns $(3, G)$, $(5, G)$, $(10, G)$, and $(12, G)$, and similar rows. These are only functions of the g and h classes. All the other functions in these rows and columns are degenerate or realizable by two boxes.

We deduce, therefore, that classes g and h can be realized by the arrangement of Fig. 7, but not class f. It is arguable that class f is more difficult to realize because it contains the symmetric functions $A+B+C=0$, 3^3 and $A+B+C=1$, 2. The latter is the function shown in Table III.

Table IV shows typical members of classes g and h, produced by the alternative arrangement of boxes. They have been constructed and numbered to give functions identical to those in Table III.

FUNCTIONS WHICH REQUIRE MORE THAN THREE BLACK BOXES

The number of functions still unrealized is 24. If the charts of Figs. 5 and 8 are compared with the symmetric functions in Fig. 3, it will be found that the symmetric functions

$$\left.\begin{aligned} A+B+C &= 0,\ 1 \\ A+B+C &= 2,\ 3 \end{aligned}\right\} \text{class } j,$$

and

have not yet been realized. These are easily seen to belong to a single symmetry class. Also the symmetric functions

$$\left.\begin{aligned} A+B+C &= 1 \\ A+B+C &= 2 \\ A+B+C &= 0,\ 2,\ 3 \\ A+B+C &= 0,\ 1,\ 3 \end{aligned}\right\} \text{class } k,$$

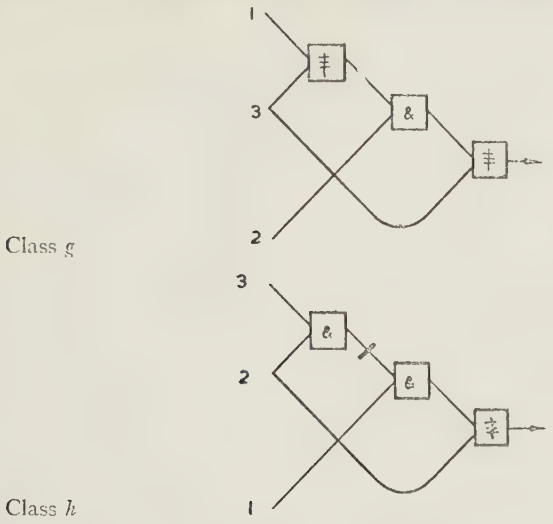
have not yet been realized, and obviously these belong to a single symmetry class, which is different from the other. These classes will be called j and k, respectively.

If these classes are plotted on the 16×16 chart, as in Fig. 9, it is found that all the remaining 24 squares are filled. Therefore, the list of classes is complete. The question of how many black boxes are needed to realize the members of classes j and k is answered by Table V, which shows typical members realized by four boxes. These arrangements were obtained by trial and error because enumeration of all the functions realizable by four boxes would be lengthy.

[3] This is short expression for "The function which takes the value 1 when $A+B+C=0$ or 3."

TABLE IV

REALIZATION OF CLASSES *g* AND *h* WITH THE ARRANGEMENT OF FIG. 7



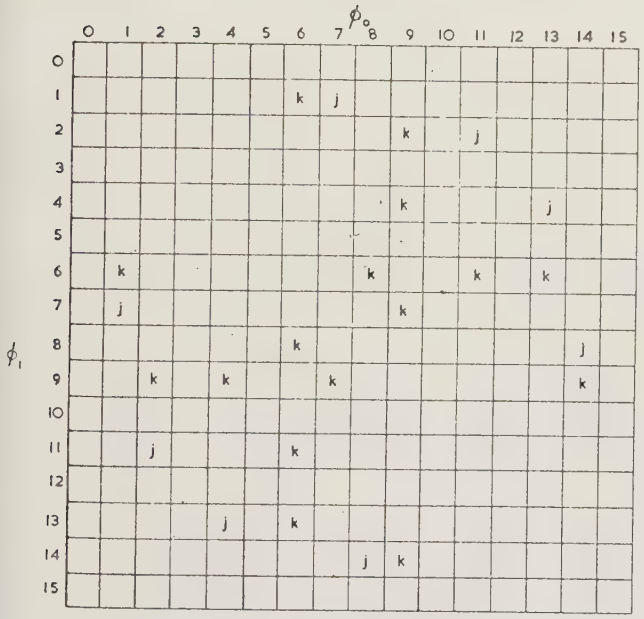


Fig. 9—The symmetry classes *j* and *k*.

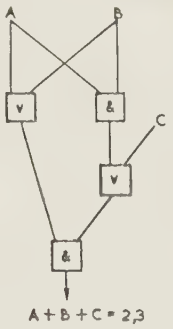
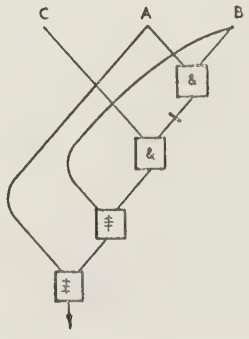
THE SYMMETRY OF THE CLASSES

The symmetry, exhibited by the ten symmetry classes described above, (and the four symmetry classes of the degenerate functions) is obviously related to the 32 point groups known to crystallographers.[4] Not all the symmetry classes have different symmetry. Classes *e* and *h* have the same symmetry, a single plane of reflection which corresponds to symmetry in two of the variables. Classes *b* and *k* have the same symmetry, a trigonal axis with three intersecting planes which correspond to symmetry in three variables. It happens that all the remaining classes have different symmetry, making 12

[4] Hilton, "Mathematical Crystallography," Oxford University Press, New York, N. Y.; 1903.

TABLE V

THE SYMMETRY CLASSES REALIZABLE BY FOUR "BLACK BOXES"

Symbol Used for Class	Typical Member	Order of Symmetry Group	No. in Class
j		12	8
k		6	16

types of symmetry altogether. By no means all the sorts of symmetry which are conceivable actually occur. For example there is no completely unsymmetric case.

Symmetry of a type not considered in the point groups occurs in classes *a*, *c*, *g*, and *j* (and two degenerate ones). Because we allowed negation of output as a symmetry operator, relations such as

$$F(\sim A, B, C) = \sim F(A, B, C)$$

are regarded as symmetry. This is analogous to a plane of symmetry such as

$$F(\sim A, B, C) = F(A, B, C)$$

and we can call it "a plane of the negation kind." There are also diagonal axes of the negation kind, as well as tetragonal axes of the negation kind which are diagonal axes of the usual kind. Inversion of the negation kind occurs as a symmetry operator in classes *a* and *j*. This is simply

$$F(\sim A, \sim B, \sim C) = \sim F(A, B, C).$$

Classes *a* and *j* also have hexagonal rotary reflection axes of the negation kind which are, as well, trigonal axes of the usual kind. All the new types of symmetry are shown in stereographic projection in Fig. 10, using the notation of Hilton's book with operators of the negation kind shown dotted.

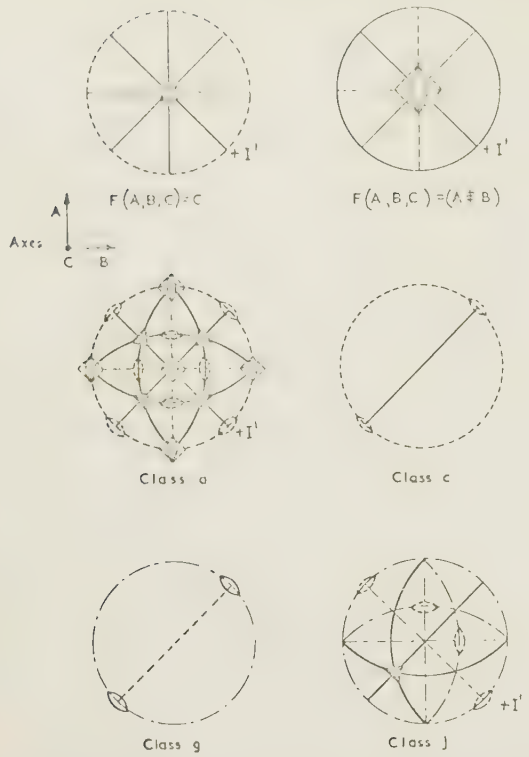


Fig. 10—Types of symmetry with operators of the negation kind. The outer circle in class g and the dashed lines in class j do not represent planes of symmetry. By I' is meant inversion of the negation kind. The trigonal axes in a and j are also hexagonal rotatory reflection axes of the negation kind.

Arrangements of Black Boxes Which Give Any Function of Three Variables

We have demonstrated that, using four black boxes only, any of the functions of three variables can be produced. To achieve this different interconnections were employed for each function, and the least number of boxes was used for each function.

The question now to be studied is whether any single arrangement of four black boxes is capable of producing (by assigning suitable functions to each box) any of the 256 functions of three variables. We shall still allow permutations of the three inputs. It will be shown that there are several such arrangements, and they will be enumerated. We shall call these arrangements which (with permutation of inputs) can represent any functions of their inputs "comprehensive" arrangements.

The Nontrivial Arrangements of Four Black Boxes

It is easy to enumerate the possible arrangements of four boxes. Arrangements in which the output of a box influences its input (*i.e.*, feedback) are not allowed. Some arrangements have a section containing more than one box which, when isolated from the rest is seen to have only two inputs and one output. This section can be reduced to a single box, so in this case, the arrangement is equivalent to one with less than four boxes and need not be considered.

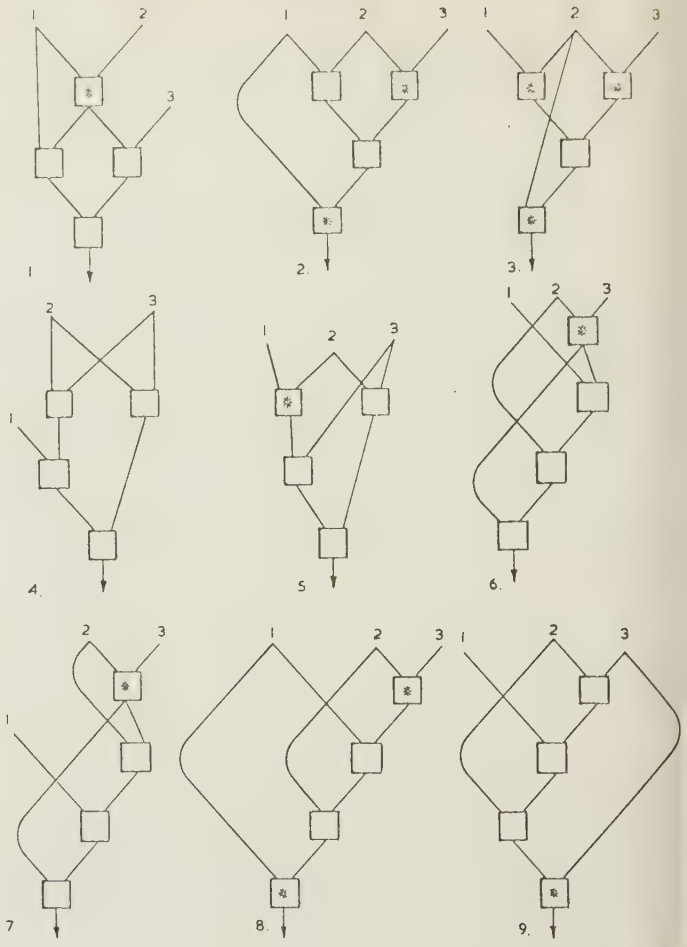


Fig. 11—The arrangements of four "black boxes."

When arrangements of these sorts have been eliminated, the nine arrangements shown in Fig. 11 are found. The details of the enumeration will not be given. (The asterisks are explained later.)

The Function of Classes a to h

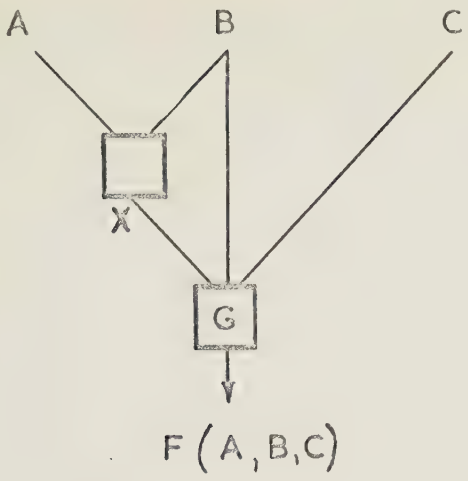
The functions of classes a to h can be realized by less than four black boxes. One way of realizing them with the arrangements of Fig. 11 is to allow one or more boxes to become trivial functions, so that the arrangements reduce to 3-box and 2-box arrangements.

There are two arrangements of three boxes, the ones shown in Figs. 6 and 7, and the arrangement of Fig. 6 covers all the classes a to h. The arrangement of Fig. 7 covers classes a to h except for class f.

It can easily be seen that, by making the appropriate box degenerate, the arrangements 1 to 7 of Fig. 11 can be made into the arrangement of Fig. 6. For these, therefore, only classes j and k remain to be investigated.

The arrangements 8 and 9 cannot degenerate to that of Fig. 6, but they do go into Fig. 7. For these arrangements, therefore, classes f, j, and k remain to be investigated.

The classes to be investigated, f, j, and k, all contain symmetrical members:



$$F(A,B,C) = G\{X(A,B), B, C\}$$

Fig. 12—A way of breaking down a complex arrangement.

for f, $A + B + C = 0, 3$ or $1, 2$;

for j, $A + B + C = 0, 1$ or $2, 3$;

for k, $A + B + C = 1$, or 2, or $0, 2, 3$ or $0, 1, 3$.

This makes investigation easier because permutations of inputs need not be considered.

Methods for Detaching One Box from an Arrangement

Suppose that an arrangement of black boxes with three inputs and one output can be represented by Fig. 12.

Suppose also that $F(A, B, C)$ has this property: whatever the value of B, there is some value of C for which F depends on A. Then whatever the value of B, X depends on A. There are only the following possibilities:

$$\begin{cases} X = A \\ X = (A \not\equiv B) \text{ and their negations.} \end{cases}$$

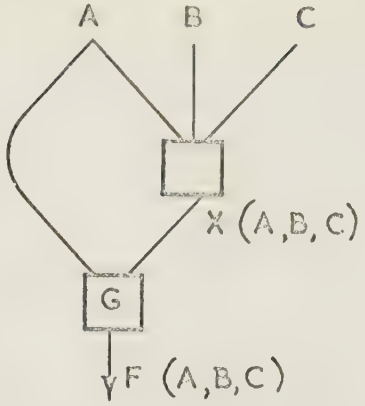
Table VI shows that functions of classes f, j, and k have this property.

TABLE VI

Class	Function	$B=0$	$B=1$
f	$A+B+C=0, 3$	$\sim(A \vee C)$	$A \,\&\, C$
j	$A+B+C=2, 3$	$A \,\&\, C$	$A \vee C$
k	$A+B+C=1$	$A \not\equiv C$	$\sim(A \vee C)$

The arrangements 1, 2, 3, 5, 6, 7, and 8 of Fig. 11 can be represented by Fig. 12, (arrangement 3 in two ways). Therefore, in representing functions of classes f, j, and k, the starred boxes on the upper levels of these arrangements can be assumed to be $\not\equiv$ boxes. The degenerate case $x = A$ is of no interest.

Suppose now that an arrangement of black boxes can be represented in the form of Fig. 13.



$$F(A,B,C) = G\{A, X(A,B,C)\}$$

Fig. 13—A further way of breaking down a complex arrangement.

Suppose also that $F(A, B, C)$ has this property: whatever the value of A, $F(A, B, C)$ can take both the values 0 and 1, depending on B and C. Then whatever the value of A, $G(A, X)$ depends on X. Again the only useful case is $G(A, X) = (A \not\equiv X)$.

The functions shown in Table VI have this property and the arrangements numbers 2, 3, 8, and 9 of Fig. 11 can be represented by Fig. 13. Therefore, in representing functions of classes f, j, and k, the starred boxes in the lower levels of these arrangements can be assumed to be $\not\equiv$ boxes.

Having found the nature of these boxes, the problem of whether a given $F(A, B, C)$ can be represented by the whole array can be reduced to the problem of representing a different function by the smaller array of unstarred boxes.

Take the case of Fig. 12. Since $X = (A \not\equiv B)$, it follows that $A = (X \not\equiv B)$. Hence $G(X, B, C) = F(A, B, C) = F(X \not\equiv B, B, C)$ is represented by the lower part.

In the case of Fig. 13, since $F = (A \not\equiv X)$, it follows that $X = (A \not\equiv F)$. Hence $X(A, B, C) = A \not\equiv F(A, B, C)$. So F is represented by the whole array if $A \not\equiv F$ is represented by the upper part.

In this way the problem is reduced in complexity, but it must be borne in mind that in the modified functions we are not allowed to interchange the inputs. This would alter the array.

Only one of the nine arrangements is unaffected by this method of reduction, namely arrangement 4. Fortunately, this has been found to be comprehensive by other methods.

Demonstration of the Comprehensive Arrangements

The following results concerning arrangements 1 to 9 were discovered by the method of reduction described in the previous section. It will be remembered that all the arrangements can, by degeneration to a three box arrangement, produce functions of classes a to e, g, and

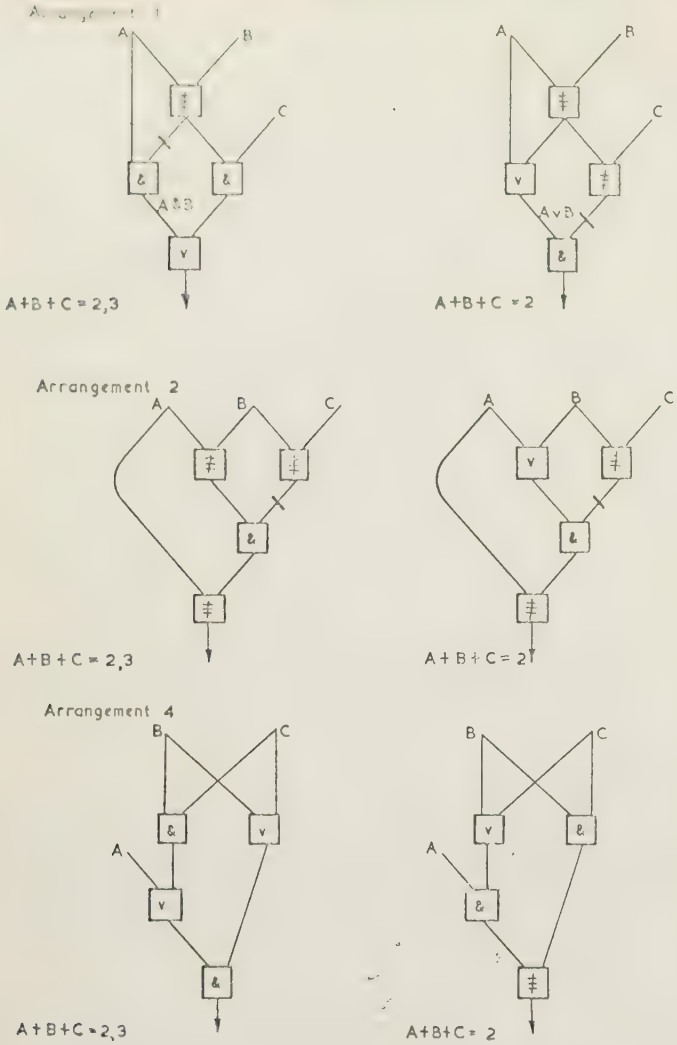


Fig. 14—The "comprehensive" arrangements.

h, and that all except arrangements 8 and 9 can produce functions of class f.

Arrangements 1, 2, and 4 are comprehensive, and their representation of classes j and k are shown in Fig. 14. The other arrangements fail in the following classes.

Arrangement	Classes
3	k
5	j
6	j, k
7	j
8	f, j, k
9	f, j

As an example of the method of reduction, consider arrangement 9 as a possible method of realizing the function of class j:

$$A + B + C = 2, 3.$$

This is an example of the type of Fig. 13, where A is the input marked 3 in Fig. 11.

If $A = 0$, $F = B\&C$, and if $A = 1$, $F = B \vee C$. Both can take the values 0 and 1, so the starred box can be taken to be a $\not\equiv$ box. Therefore, the unstarred boxes represent the function

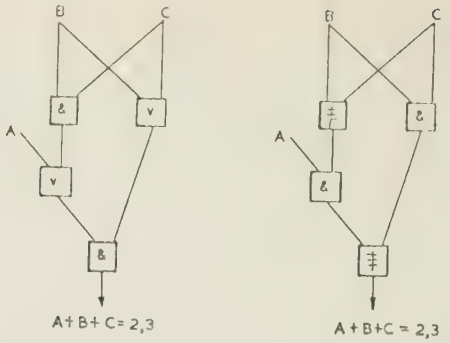


Fig. 15—Alternative realizations of a function.

$$X = (A \not\equiv F),$$

for which if $A = 0$, $X = B\&C$; and if $A = 1$, $X = \sim(B \vee C)$, This is seen, from Table VI, to be a function of class f. hence not realizable by the three boxes (see Table IV).

All the arrangements except 1, 2, and 4 give similar results.

There are, therefore, three comprehensive arrangements of four black boxes, that is, arrangements which can generate any function of three variables by a suitable choice of boxes. The choice of boxes is not unique. For example, the two choices in Fig. 15 give the same output.

Comprehensive Arrangements without Allowing Permutations of Inputs

The final problem to be discussed is: which arrangements of four black boxes are capable of producing all 256 functions of three variables without allowing permutations of inputs? Such an arrangement must evidently be comprehensive.

Consider the function $F(A, B, C) = (A\&B) \not\equiv C$. We assert that the arrangements of Fig. 16 cannot produce this function. For, by applying the argument of an earlier section, we find that the boxes marked with a star in each case must be either $\not\equiv$ or degenerate. The degenerate cases, shown in Fig. 17, are easily shown to be impossible by further reduction. The nondegenerate case of Fig. 16(a) gives a new function for the unstarred boxes,

$$G(A, B, X) = F(A, B, X \not\equiv A)$$
$$= (A \,\&\sim B) \not\equiv X.$$

Fig. 17(a) must represent this function with X replacing C. Further reduction shows this to be impossible.

The nondegenerate case of Fig. 16(b), gives the arrangement of Fig. 18, where

$$Y(A, X, C) = A \not\equiv F(A, C \not\equiv X, C)$$
$$= (\sim A \,\&\, C) \vee (A \,\&\sim X).$$

This is a function of class g and requires three black boxes.

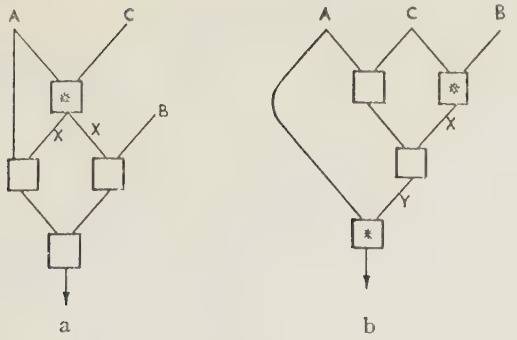


Fig. 16

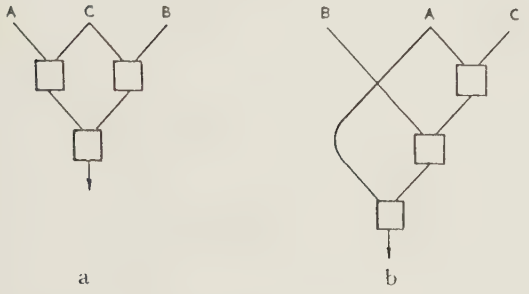


Fig. 17

The only remaining comprehensive arrangement, which is number 4, is capable of giving any function of three variables without permutation of inputs. For suppose an arbitrary $F(A, B, C)$ is specified by giving

$$\phi_0(A, B) = F(A, B, 0)$$

and

$$\phi_1(A, B) = F(A, B, 1)$$

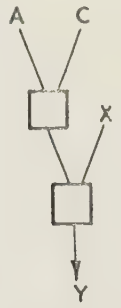


Fig. 18

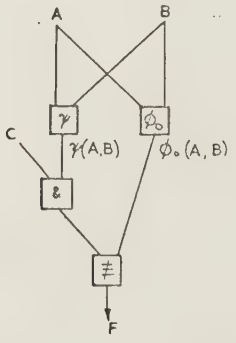


Fig. 19—A completely general arrangement with four boxes.

and let $\psi(A, B) = \phi_0(A, B) \neq \phi_1(A, B)$. Then the arrangements of Fig. 19 gives $F(A, B, C)$.

Acknowledgment

The author wishes to acknowledge the help of M. Woodger in making useful suggestions and reading the manuscript.

Analysis of Sequential Machines*

D. D. AUFENKAMP† AND F. E. HOHN‡

Summary—This paper begins with Mealy's model of a sequential machine and introduces a "connection matrix" which describes the machine completely. The "equivalence" of states of such a machine may be analyzed systematically by an iterative technique, the validity of which is rigorously established. Once equivalence is completely analyzed, it is a simple matter to write the connection matrix for the simplest equivalent machine. The process is not difficult to execute, even in complex cases, and could be programmed for a computer.

I. Sequential Machines

ONE may designate as a "sequential machine"any device which produces prescribed sequences of outputs in response to given sequences of inputs.

A number of mathematical models of such machines have been proposed.[1–5] Mealy's model of a sequential machine[1] assumes that the machine may be described in terms of

1) a finite number of states $s_1, s_2, \cdots, s_n$,
2) a finite number of distinct *inputs* symbolized by $x_1, x_2, \cdots, x_m$, and
3) a finite number of distinct *outputs* symbolized by $y_1, y_2, \cdots, y_p$. It is assumed that the *present output* and the *next state* are uniquely determined by the *present state* and the *present input*.

Each such machine M may now be represented by a weighted, directed graph or net,[6,7] called the *state diagram* of M. The set of vertices $v_1, v_2, \cdots, v_n$ of the net corresponds in 1-1 fashion to the set of states $s_1, s_2, \cdots, s_n$ of M and the s_j's are the weights associated with these vertices. For each input which effects a transition of the machine from state s_i to state s_j, there is a directed branch from the vertex v_i to the vertex v_j and we assign this branch as its weight, the ordered pair consisting of the input symbol in question and the symbol for the corresponding output. When there is more than one such branch, say, k branches from a vertex v_i to a vertex v_j, we simplify matters by employing only one branch and assigning it a weight $(x_{r_1}, y_{r_1}) \vee (x_{r_2}, y_{r_2}) \vee \cdots \vee (x_{r_k}, y_{r_k})$ (where $\vee$ means "or") which defines the union of the weights of all the original branches from v_i to v_j.

A sequential machine satisfying the above requirements will be referred to simply as a *machine* in the rest of this paper since no other types of sequential machines are discussed.

It is often convenient to identify the states simply as $1, 2, \cdots, n$ and to use unsubscripted symbols for inputs and outputs. We illustrate in Fig. 1 by giving a state diagram of a binary pulse-divider. The states are 1 and 2, the input symbols are 0 and 1, and the output symbols are 0 and 1 also. Here the input sequence,

$$1, 1, 1, \cdots$$

starting with the machine in state 1, produces the output sequence

$$0, 1, 0, 1, \cdots.$$

However, no matter what the input sequence may be, if the machine starts in state 1, every second 1 in the input sequence will result in an output 1, all other outputs being 0's.

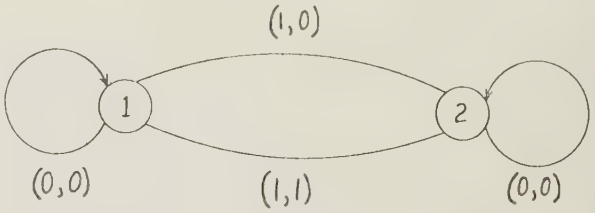


Fig. 1—State diagram of a binary pulse-divider.

II. The Connection Matrix

The basic tool of analysis in what follows is the *connection matrix* of the net associated with the machine. We recall its definition from Hohn, *et al.*[6]

$$C = [c_{ij}]_{n \times n}$$

where c_{ij} is the above defined union ($\vee$) of the branch weights on all branches from vertex i to vertex j. If no branches connect vertex i to vertex j, then $c_{ij} = 0$. No input symbol will appear more than once in any row of C since the next state is uniquely determined by the present state and the present input, *i.e.*, since M is a deterministic machine.

The connection matrix of the binary pulse-divider illustrated in Fig. 1 is

$$C = \begin{bmatrix} (0,0) & (1,0) \\ (1,1) & (0,0) \end{bmatrix}.$$

* Manuscript received by the PGEC, July 17, 1957; revised manuscript received, October 2, 1957.

† Missile Systems Division, Lockheed Aircraft Corp., Palo Alto, Calif.; formerly at the University of Illinois, Urbana, Ill.

‡ University of Illinois, Urbana, Ill.

[1] G. H. Mealy, "A method for synthesizing sequential circuits," *Bell Sys. Tech. J.*, vol. 34, pp. 1045–1079; September, 1955.

[2] E. F. Moore, "Gendanken—Experiments on Sequential Machines," in "Automata Studies," Princeton University Press, Princeton, N.J., pp. 129–153; 1956.

[3] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, pp. 161–190, 275–303; March and April, 1954.

[4] A. W. Burks and H. Wang, "The Logic of Automata," Univ. of Michigan, Ann Arbor, Mich., Eng. Res. Inst.; 1956.

[5] J. A. Caupeau, "The synthesis and analysis of digital systems by Boolean matrices," this issue, pp. 231–241.

[6] F. E. Hohn, S. Seshu, and D. D. Aufenkamp, "The theory of nets," IRE Trans., vol. EC-6, pp. 154–161; September, 1957.

[7] Mealy, *op. cit.*, p. 1052.

III. Input-Output Polynomials

In what follows, we shall frequently find it necessary to employ certain $\vee$ polynomials in the input-output pairs associated with a machine. We now define these polynomials and list their basic properties.

In terms of the notation of Section 1, the possible input-output pairs (x_i, y_j) are mp in number. Let us denote them by $P_1, P_2, \cdots, P_{mp}$. From the P's we then form input-output polynomials with the aid of finitely many operations of multiplication (indicating succession) denoted in the usual way, and union (the logical "or") denoted by "$\vee$". Concerning the set S of such polynomials, we now make the following postulates, in which "$=$" denotes identity and f, g, h are arbitrary members of S:

1) $P_1, P_2, \cdots, P_{mp}$ belong to S,
2) $f \vee g$ belongs to S,
3) $f \vee g = g \vee f$,
4) $f \vee f = f$,
5) $(f \vee g) \vee h = f \vee (g \vee h)$,
6) fg belongs to S.

There exists in S an input-output polynomial "0" (not to be confused with a possible input symbol 0) such that

7) $f \vee 0 = f$ and
8) $f \cdot 0 = 0 \cdot f = 0$,
9) If $f \neq 0$, $g \neq 0$, $f \neq g$, then in general, $fg \neq gf$,
10) $f(gh) = (fg)h$,
11) $(f \vee g)h = fh \vee gh$ and
$f(g \vee h) = fg \vee fh$.

A polynomial $P_{i_1} P_{i_2}, \cdots, P_{i_r}$ will be called a *term*. We define an *expanded polynomial* of S to be a term or a union of terms. Every polynomial of S may be expressed as an expanded polynomial with the aid of the above postulates. If each term of an expanded polynomial f of S is a product of r factors, the polynomial will be said to be *homogeneous of degree r*. The polynomial 0 is assigned no degree.

The above postulates allow us also to define a useful form of matrix multiplication. Consider two matrices $A_{\alpha \times \beta}$ and $B_{\beta \times \gamma}$ whose entries are members of S. We define AB in the usual way except that "$\vee$" replaces "$+$."

$$AB = [a_{i1}b_{1j} \vee a_{i2}b_{2j} \vee \cdots \vee a_{i\beta}b_{\beta j}]_{\alpha \times \gamma}$$

or, more compactly,

$$AB = \left[\bigvee_{k=1}^{\beta} a_{ik}b_{kj} \right].$$

Now let C be the connection matrix of a machine M. Then, from the definition of C it follows that

$$C^2 = \left[\bigvee_{k=1}^{\beta} c_{ik}c_{kj} \right]$$

is a matrix whose ij entry gives all *input-output sequences of length* 2 which take M from state i to state j via some intermediate state k (not necessarily distinct from i or j). More generally we have

$$C^r = \left[\bigvee_{k_1, k_2, \cdots, k_{r-1}=1}^{n} c_{ik_1}c_{k_1k_2} \cdots c_{k_{r-2}k_{r-1}}c_{k_{r-1}j} \right]$$

the ij entry of which gives all *input-output sequences of length r* which take M from state i to state j via $r-1$ intermediate states.

The nonzero entries of C^r will be homogeneous polynomials of degree r. Moreover, since the next state is uniquely determined by the present state and the present input, no two entries in the same row of C^r can have terms involving the same input sequence.

It is also useful to define the *union of two matrices* $A_{\alpha \times \beta}$ and $B_{\alpha \times \beta}$ as the matrix $A \vee B$ defined by

$$A \vee B = [(a_{ij} \vee b_{ij})]_{\alpha \times \beta}.$$

The union has the properties

$$A \vee B = B \vee A,$$
$$(A \vee B) \vee C = A \vee (B \vee C),$$

of which we shall make use.

IV. Input Restrictions

It is possible that in the state diagram there will be vertices such that not every input symbol is associated with a branch leaving that vertex. In this case, when the machine is in the state corresponding to this vertex, its output and next state are not defined for every input and we say that the machine has *input restrictions*. If there are no input restrictions, the machine must be capable of responding to any input no matter in which state it may be so that at each vertex, each input symbol will appear on some branch leaving that vertex.

To illustrate, suppose there are three input symbols a, b, c and that the machine contains a state i such as is shown in Fig. 2. Here the machine would not be able to accept an input a if it were in state i.

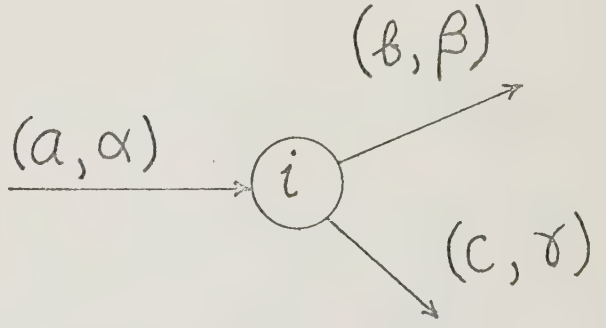


Fig. 2—Example of an input restriction associated with state i.

The presence of input restrictions is related in practice to the problem of designing a machine which responds in a prescribed way to a limited number of *input sequences* (*i.e.*, sequences of input symbols). Then the restrictions which define the given set of *input sequences* are reflected in the resulting state diagram. This problem will be treated in detail in a later paper of this series.

At the moment, our concern is the analysis of a given state diagram. For this purpose the next definition is useful.

Any input sequence which, starting with the machine in state i, violates no input restriction of state i or any subsequent state, is called *an allowable input sequence for state i.*

We have noted in the last section that the ij entry of C^r lists all sequences of r input-output pairs which take the machine from state i to state j. Neglecting the output symbols, we have then in the ith row simply the allowable input sequences of length r for state i. (If only input sequences are of interest, we omit the output symbols entirely from C and compute powers of this simpler matrix. When this is done, one must distinguish at times between an input symbol "0" and the 0 polynomial mentioned above.)

If no input restrictions are present, every input sequence may be presented to M regardless of its initial state and hence every input sequence of length r must appear somewhere in each row of C^r.

V. Synchronous and Asynchronous Machines

The notion of input restrictions enables us to draw an important distinction. The basic characteristic of synchronous machines is that with the aid of clocking devices they are enabled to distinguish between successive, identical inputs. This suggests the following definitions.

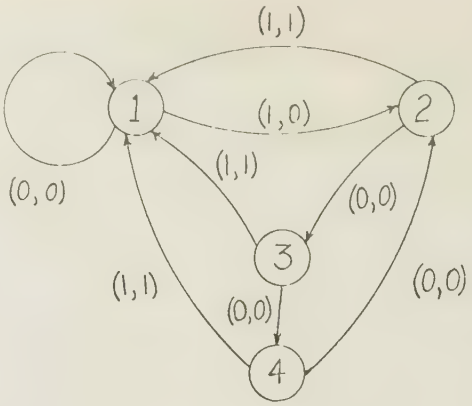
If the allowable sequences for the various states of a given machine M include at least one sequence in which an input symbol follows itself, M will be called *essentially synchronous.* If no such sequence is included, *i.e.*, if an input symbol never follows itself in an allowable input sequence, M will be called *essentially asynchronous.*

When we say "essentially asynchronous," we do not mean to imply that the machine may not be realized with synchronous circuitry. We mean only that the machine is never required to distinguish between successive, identical inputs. Thus the input sequence in Fig. 3 would be recognized by an essentially asynchronous machine as 0 1 0 whereas an essentially synchronous machine might recognize it as 0 1 1 0 0 0, for example.

Fig. 3—Input sequence.

VI. Equivalent States and Equivalent Machines

In problems of analysis and synthesis of sequential machines, we often wish to determine whether or not a given state diagram can be simplified, that is, whether or not it can be replaced by one having fewer states without altering any of the relations between input and output sequences which characterize the machine.



$$C = \left[\begin{array}{c|ccc} (0,0) & (1,0) & 0 & 0 \\ \hline (1,1) & 0 & (0,0) & 0 \\ (1,1) & 0 & 0 & (0,0) \\ (1,1) & (0,0) & 0 & 0 \end{array}\right]$$

Fig. 4—State diagram and associated connection matrix of a machine having two inputs and two outputs in which states 2, 3 and 4 are equivalent.

The notions of equivalent states and equivalent machines, which ultimately go back to Moore[2] and Huffman,[3] are essential to the solution of this problem. We therefore make the following definitions.

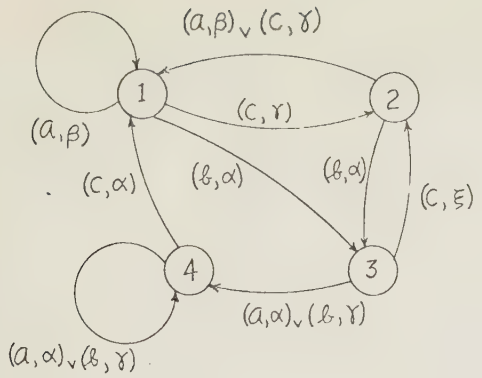
Definition 1: Two states, s_i of a machine M and t_j of a machine N, are called *equivalent states* if, and only if, they have the same set of allowable input sequences and, starting with M in s_i and N in t_j, the two machines will yield identical output sequences on being presented with the same allowable input sequence. In this definition M and N may of course be the same machine.

Definition 2: Two machines M and N are called *equivalent machines* if and only if for each state s_i of M there exists at least one equivalent state t_j of N and for each state t_i of N there exists at least one equivalent state s_j of M.

The relations of "equivalence" here defined are true equivalence relations. It is in fact easy to check that the determinative reflexive, symmetric, and transitive properties hold in each case.

As an example, consider the machine with inputs 0, 1 and outputs 0, 1 whose state diagram and connection matrix are shown in Fig. 4. Here every input sequence is allowable for every state.

Now for two states i and j to be equivalent, it is necessary that the same input-output pairs appear in rows i and j of C. (That this condition is not sufficient is illustrated in the example in Section I. In the above example, it is therefore possible that states 2, 3, and 4 are equivalent. The input 1 takes each of states 2, 3, 4



$$\begin{bmatrix} (a,\beta) & (c,\gamma) & (b,\alpha) & 0 \\ (a,\beta)\vee(c,\gamma) & 0 & (b,\alpha) & 0 \\ 0 & (c,\xi) & 0 & (a,\alpha)\vee(b,\gamma) \\ (c,\alpha) & 0 & 0 & (a,\alpha)\vee(b,\gamma) \end{bmatrix}$$

Fig. 5—Machine with three inputs a, b, c and three outputs α, β, γ of which ξ is one.

into state 1, yielding the output 1 in each case. The input 0 simply permutes these states among themselves, now yielding the output 0 in each case. By inspection of the connection matrix we therefore see that any input sequence will result in the same output sequence whether we start with the machine in state 2, 3, or 4. We now replace the class of equivalent states 2, 3, 4 by a single state 2′ which is taken by input 1 into a state 1′ and by input 0 into state 2′, the outputs being 1 and 0, respectively. Input 0 takes 1′ into 1′; input 1 takes 1′ into 2′. This yields a state diagram essentially the same as that previously shown in Fig. 1. The connection matrix of this machine, namely

$$\begin{bmatrix} (0,0) & (1,0) \\ (1,1) & (0,0) \end{bmatrix}$$

is readily obtained by replacing each submatrix of the partitioned matrix of Fig. 4 by the union of all its entries. This observation will presently be reduced to a formal rule.

It remains to point out that the original machine and the new machine are equivalent, the correspondence of states being

$$1 \leftrightarrow 1' \quad \text{and} \quad (2, 3, 4) \leftrightarrow 2'.$$

In fact, for state 1(1′) the input 0 yields output 0 and a transition to state 1(1′) but input 1 yields output 0 and a transition to state 2(2′). For states 2, 3, 4(2′) input 0 yields output 0 and a transition to state 3, 4, 2(2′) while input 1 yields output 1 and a transition to state 1(1′). Thus, starting with corresponding initial states, both machines will produce identical output sequences for a given input sequence, and hence are equivalent.

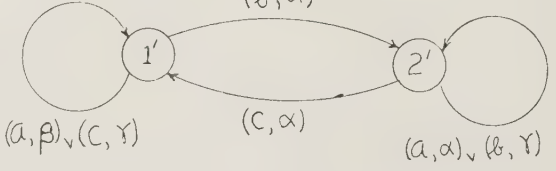
VII. Additional Examples

We now give two additional examples to illustrate certain aspects of equivalence before attacking the problem formally. Consider first the state diagram and associated connection matrix shown in Fig. 5. The machine is assumed to have 3 inputs a, b, c and 3 outputs α, β, γ of which ξ is one. There are no input restrictions.

First we consider states 1 and 2, noting that rows 1 and 2 contain the same input-output pairs. Hence equivalence of these two states is possible. Input a takes both states into state 1 whereas b takes both into state 3. Input c on the other hand merely permutes states 1 and 2. From these several observations we see that, starting with the machine in state 1 or state 2, a given input sequence will result in the same output sequence in either case. Hence 1 and 2 are equivalent states.

Now examine states 3 and 4 under the assumption that $\xi=\alpha$ in which case rows 3 and 4 contain identical input-output pairs. The input c then takes 3 and 4 into states *already known to be equivalent* whereas inputs a and b take 3 and 4 into state 4. It again follows that states 3 and 4 are equivalent. On the other hand, no further equivalences are possible because of different input-output pairs in the rows.

We now replace each submatrix of the partitioned C matrix by the union of all its entries, thus obtaining the 2×2 matrix and associated state diagram shown in Fig. 6.



$$C' = \begin{array}{c} \begin{matrix} 1' & \quad 2' \end{matrix} \\ \begin{bmatrix} (a,\beta)\vee(c,\gamma) & (b,\alpha) \\ (c,\alpha) & (a,\alpha)\vee(b,\gamma) \end{bmatrix} \end{array} \begin{matrix} 1' \\ 2' \end{matrix}$$

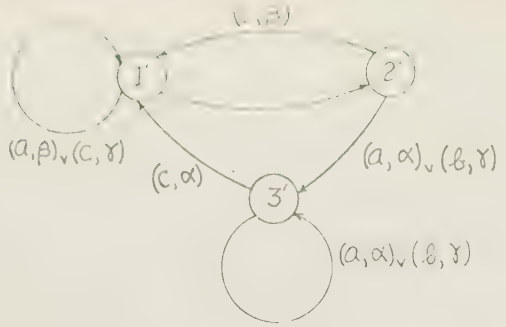
Fig. 6—The machine given in Fig. 5 when reduced under equivalence of states assuming that $\xi=\alpha$.

As in the example of the preceding section, one can verify that this machine is equivalent to the originally given one, the correspondences between states being

$$(1, 2) \leftrightarrow 1' \quad \text{and} \quad (3, 4) \leftrightarrow 2'.$$

Finally, suppose that $\xi\neq\alpha$, say $\xi=\beta$. Then states 3 and 4 cannot be equivalent, because of the different outputs associated with input c. In this case we partition C further:

$$C' = \left[\begin{array}{c|c|c|c} (a,\beta) & (c,\gamma) & (b,\alpha) & 0 \\ (a,\beta)\vee(c,\gamma) & 0 & (b,\alpha) & 0 \\ \hline 0 & (c,\beta) & 0 & (a,\alpha)\vee(b,\gamma) \\ \hline (c,\alpha) & 0 & 0 & (a,\alpha)\vee(b,\gamma) \end{array}\right]$$



$$C' = \begin{array}{c} \begin{matrix} 1' & 2' & 3' \end{matrix} \\ \begin{bmatrix} (\alpha,\beta)\vee(C,\gamma) & (\beta,\alpha) & 0 \\ (C,\beta) & 0 & (\alpha,\alpha)\vee(\beta,\gamma) \\ (C,\alpha) & 0 & (\alpha,\alpha)\vee(\beta,\gamma) \end{bmatrix} \end{array} \begin{matrix} 1' \\ 2' \\ 3' \end{matrix}$$

Fig. 7—The machine given in Fig. 5 when reduced under equivalence of states assuming that $\xi=\beta$.

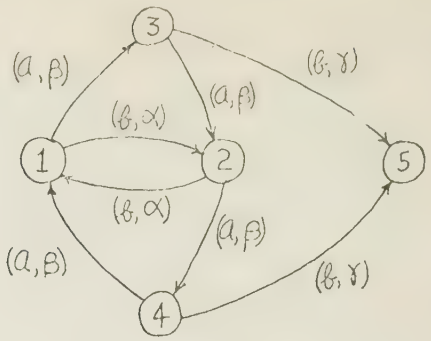
so that the rows of each submatrix correspond to equivalent states and similarly for columns. Again, we replace each submatrix by the union of all its entries and thus obtain the connection matrix of a demonstrably equivalent machine with fewer states, as shown in Fig. 7.

The type of equivalence exemplified by states 1 and 2 above, and by 3 and 4 when $\xi=\alpha$, is readily generalized. Suppose $i_1, \cdots, i_m$ are states such that the same set of input-output pairs appears in each of the corresponding rows of C. If *each* input x_{j_r} of the set of inputs $x_{j_1}, \cdots, x_{j_k}$ takes all of the states $i_1, \cdots, i_m$ into a fixed state i_s, or into states equivalent to i_s and if the *remaining* inputs to the machine simply map the set $i_1, \cdots, i_m$ onto or into itself, then the states of this set are equivalent. More generally, there may be a number of disjunct sets of inputs as the set $x_{j_1}, \cdots, x_{j_k}$.

These conclusions will be justified by the theorems to be proved in Section VIII. A corresponding partitioning of the connection matrix, analogous to that used in the above example, may be effected by the methods of Section IX.

As a second example, consider the state diagram and matrix shown in Fig. 8. The inputs are a, b; the outputs are α, β, γ. The input sequences are restricted since the machine will not respond to either input once it is in state 5. (We call state 5 a *terminal state* of the machine.)

State 5 is obviously equivalent to no other state here. However, states 1 and 2 are permuted by input b while input a takes them into states 3 and 4, respectively. Since the input-output pairs in rows 1 and 2 are the same, we conclude from the generalization of the previous example that 1 and 2 will be equivalent if 3 and 4 are. Now input a takes 3 and 4 into states 2 and 1 respectively while input b takes them into a common next state, namely state 5. Thus we would conclude that 3 and 4 are equivalent if 1 and 2 are! However, examination of the possible input-output sequences reveals that 1 and 2 are indeed equivalent, as are 3 and 4.



$$C = \left[\begin{array}{c|c|c|c|c} 0 & (\beta,\alpha) & (\alpha,\beta) & 0 & 0 \\ (\beta,\alpha) & 0 & 0 & (\alpha,\beta) & 0 \\ \hline 0 & (\alpha,\beta) & 0 & 0 & (\beta,\gamma) \\ (\alpha,\beta) & 0 & 0 & 0 & (\beta,\gamma) \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Fig. 8—Machine having a terminal state in which there exist equivalent states.

As before, we now replace each submatrix by the union of its entries and obtain the connection matrix of a demonstrably equivalent, 3-state machine, as shown in Fig. 9, opposite.

The type of equivalence illustrated here may also be generalized. Suppose that the rows of C corresponding to a set of states S_1 all contain the same input-output symbols. Similarly for S_2, S_3, $\cdots$, S_r, where these are *disjoint* sets of states. If each input x_i to the machine takes *all* the states of each S_j into the same state, or into equivalent states, or into states all contained in some S_k, then each S_j contains only equivalent states. Again, this type of equivalence is identified by partitioning the connection matrix in a manner analogous to that illustrated in the above example. The correctness of this generalization will appear as a consequence of the results in the next two sections.

We are now ready to examine the problem of equivalence in general.

VIII. The Fundamental Theorems

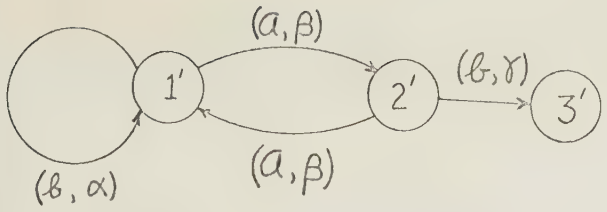
We begin with some definitions. A matrix A whose elements are input-output polynomials is called an *r matrix* if it has the following properties:

1) all nonzero entries of A are homogeneous and of a common degree r,

2) all nonzero terms in each row of A have distinct input sequences, and,

3) all nonzero terms which appear in any given row also appear in every other row. (The submatrices in the examples of the last section are 1 matrices.)

Let the columns of a matrix A of order n be partitioned into q sets of $N_1, N_2, \cdots, N_q$ columns where $\sum_{j=1}^{q} N_j = n$. If now the rows are also partitioned into



$$C' = \begin{array}{c} \begin{matrix} 1' & 2' & 3' \end{matrix} \\ \begin{bmatrix} (\beta,\alpha) & (a,\beta) & 0 \\ (a,\beta) & 0 & (\beta,\gamma) \\ 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} 1' \\ 2' \\ 3' \end{matrix}$$

Fig. 9—State diagram and associated connection matrix of a reduced machine equivalent to the one given in Fig. 8.

sets of $N_1, N_2, \cdots, N_q$ rows, such that the ordered sets of N_i column numbers are identical with the ordered sets of N_i row numbers, for $i=1, \cdots, q$, the matrix A is said to be *symmetrically partitioned.* (Examples appear in Section VII.) If a symmetrical partitioning is such that $N_1=N_2=\cdots=N_q=1$, so that $q=n$, then we call it the *trivial partitioning.* We now have:

Lemma 1:

The union of two r matrices A and B of the same order is again an r matrix whenever the terms of corresponding rows of A and B contain no identical input sequences. It it easy to verify that properties 1), 2), and 3) all still hold for the union under the stated hypotheses.

Lemma 2:

The product AB of an r matrix A and an s matrix B is an r+s matrix. Property 1) is immediate. To prove property 2), it will suffice to show that $a_{ik_1}b_{k_1j_1}$ and $a_{ik_2}b_{k_2j_2}$ contain distinct input sequences whenever $k_1 \neq k_2$ or $j_1 \neq j_2$ or both. Indeed, if $k_1 \neq k_2$, then by hypothesis the terms of a_{ik_1} and a_{ik_2} contain no common input sequences. Hence, because of the noncommutative nature of multiplication and the homogeneity of the entries of A and B, it follows that no two terms of $a_{jk_1}b_{k_1j_1}$ and $a_{ik_2}b_{k_2j_2}$ can contain common input sequences. If now $k_1=k_2$ but $j_1 \neq j_2$, the desired conclusion follows from the fact that no two terms of $b_{k_1j_1}$ and $b_{k_2j_2}$ contain the same input sequence.

To prove property 3), consider any term t appearing in the i_1th row of the product AB. Suppose t is contained in $a_{i_1k_1}b_{k_1j_1}$. Then the factor of t provided by $a_{i_1k_1}$ also appears in $a_{i_2k_2}$ for any given $i_2 \neq i_1$ and for *some* k_2. The reason is that A is an r matrix and every entry in row i appears somewhere in row k_2. Similarly, the factor provided by $b_{k_1j_1}$ also appears in $b_{k_2j_2}$ for some j_2. Hence, the term t also appears in the product $a_{i_2k_2}b_{k_2j_2}$, *i.e.*, property 3) is established.

We can now prove the first fundamental result:

Theorem 1: Let the submatrices of a symmetric partitioning of a given connection matrix C be denoted by C_{ij}, $i, j=1, 2, \cdots, q$. Let the submatrices of an identical partitioning of C^r, where r is any positive integer, be denoted by $C_{ij}^{(r)}$, $i, j=1, 2, \cdots, q$. Then if each C_{ij} is a 1 matrix, each $C_{ij}^{(r)}$ is an r matrix.

The proof is by induction on r. The theorem is clearly true for $r=1$. Assume that it holds for $r=k$. Examining the matrix C^{k+1}, partitioned identically to C and C^k, we have

$$C^{k+1} = C \cdot C^k = \left[\bigvee_{s=1}^{q} C_{is}C_{sj}^{(k)} \right].$$

By Lemma 2, each product $C_{is}C_{sj}^{(k)}$ is a $(k+1)$ matrix. Since C is a connection matrix, the 1 matrices C_{is_1} and C_{is_2} have no input symbols in common. Hence no two products $C_{is_1}C_{s_1j}^{(k)}$ and $C_{is_2}C_{s_2j}^{(k)}$ can have an input sequence in common. Therefore, by Lemma 1, the union

$$\bigvee_{s=1}^{q} C_{is}C_{sj}^{(k)}$$

is also a $(k+1)$ matrix and the theorem is proved.

We next use this result to characterize equivalent states in the connection matrix:

Theorem 2: If a connection matrix admits a symmetrical partitioning in which each submatrix is a 1 matrix, then the states contained in each set of the corresponding partitioning of the states are equivalent.

As we have seen in Section IV, each allowable but otherwise arbitrary *input* sequence of length r which may be applied with the machine beginning in state i must appear in some term in the ith row of C^r. If, on the other hand, C^r is partitioned identically to C, then each of the resulting submatrices is an r matrix, by Theorem 1. Therefore, the *output* sequences corresponding to allowable but otherwise arbitrary input sequences associated with states in a given partition are identical, independently of which state of the set the machine is in initially.

IX. An Algorithm for Determining All Sets of Equivalent States

Consider the connection matrix C of any machine M. The following iterative procedure will determine *all* sets of equivalent states of M.

1) Partition the states of M into disjunct sets $S_1, S_2, \cdots, S_q$ of maximal order such that the rows of C associated with the states in each S_i form a 1 matrix. If the resulting partitioning is the trivial partitioning (into sets of order 1) then there are no equivalent states in M.

2) If the previous partitioning is nontrivial, reorder the states according to the partitioning $S_1, S_2, \cdots, S_q$, if necessary, and partition the corresponding C sym-

metrically with respect to the S_i, indicating the partitioning by $C=[C_{ij}]$. If the C_{ij} are all 1 matrices, then the process terminates at this point because the states in each S_i are equivalent by Theorem 2.

3) If the C_{ij} are not all 1 matrices, then partition the states associated with each row of submatrices $C_{i1}, C_{i2}, \cdots, C_{iq}$, $i=1, 2, \cdots, q$ into disjunct sets $S_{i1}, S_{i2}, \cdots, S_{ih_1}$ of maximal order such that for each S_{ij} each of the associated submatrices in $C_{i1}, C_{i2}, \cdots, C_{iq}$ is a 1 matrix. If this new partitioning is trivial for all rows of submatrices $C_{i1}, C_{i2}, \cdots, C_{iq}$, then there are no equivalent states.

4) If the previous partitioning of the states is nontrivial, repartition the connection matrix with respect to the sets $S_{i1}, S_{i2}, \cdots, S_{ih_1}$, again reordering the states if necessary so as to have the rows of states in the same set S_{ij} adjacent to each other in the matrix. Denote the new partitioning by

$$C = [C_{ij}] = [D_{ij}{}^{kt}]; \qquad C_{ij} = [D_{ij}{}^{kt}],$$

where the indices k and t are used to denote the submatrices in the partitioning of each submatrix C_{ij}.

If now the $D_{ij}{}^{kt}$ are all 1 matrices, the process terminates at this point, for by Theorem 2 the states in each S_{ij} are equivalent. Otherwise we repeat steps 3) and 4).

We have thus outlined a two-step process: 1) partitioning the set of states into maximal, disjunct sets such that the submatrices formed from the corresponding sets of rows in the submatrices of the previous partitioning are 1 matrices, and 2) repartitioning the connection matrix with respect to this new partitioning of states. These two steps are repeated until a partitioning of the connection matrix is obtained in which *all* submatrices are 1 matrices. This is a finite process since the number of states is finite. Once such a partitioning is obtained, then by Theorem 2 we see that all states in each subset of states of the final partitioning are equivalent. If the subsets of states ultimately obtained are all of order 1, then there are no equivalent states.

At each stage, the partitioning of the states is uniquely defined by the requirements listed above. Thus the end result must also be unique. Indeed, what we have done is simply to divide the states of the machine into equivalence classes.

X. The Reduced Machine

After completing the process outlined in the last section, we define a new machine by replacing, in effect, the states of each equivalence class by a single state representing that class. To account for the transitions of all the states of one class to states of other classes under the various inputs to M, we replace each submatrix of the final partitioned form of C by the union of all its entries. For simplicity of notation denote the final partitioned form of C by

$$C = [C_{ij}] \qquad i, j = 1, 2, \cdots, q,$$

the corresponding sets of equivalent states being $S_1, S_2, \cdots, S_q$. Then if d_{ij} is the union of all the entries of C_{ij}, the $q \times q$ matrix

$$D = [d_{ij}]$$

is the connection matrix of a machine with q states, no states of which are equivalent. That D is indeed a connection matrix is seen by noting that because C is a connection matrix and the C_{ij}'s are 1 matrices, no two matrices C_{ij_1} and C_{ij_2} contain any common input symbols. Therefore d_{ij_1} and d_{ij_2} will contain no common input symbol, which is the only condition to be satisfied.

The machine N defined by D is called the *reduced form* of M[8,9]. It remains to show that M and N are equivalent machines.

To prove this, for each positive integer r we partition C^r according to the sets of equivalent states of M, and let $C_{ij}{}^{(r)}$ denote any of the resulting submatrices. The entries of $C_{ij}{}^{(r)}$ define the allowable input sequences of length r which take the states of set S_i into those of set S_j and also give the associated output sequences.

Now the entry d_{ij} of D is the union of the entries of C_{ij} and hence defines the set of all inputs which take some member of S_i into some member of S_j. On the other hand, $d_{ij}{}^{(r)}$, the ij entry of D^r, is given by

$$d_{ij}{}^{(r)} = \bigvee_{k_1, k_2, \cdots, k_{r-1}}{}^{r} d_{ik_1} d_{k_1 k_2} \cdots d_{k_{r-1} j}.$$

Any term in the expansion of this product represents an input-output sequence taking some state of S_i into some state of S_j. Hence, this term appears in $C_{ij}{}^{(r)}$, and because $C_{ij}{}^{(r)}$ is an r matrix, this term appears *in each row* of $C_{ij}{}^{(r)}$. Conversely, since each state of M appears in some S_i, every term appearing in $C_{ij}{}^{(r)}$ has its factors appearing in some produce of the form $d_{ik_1} d_{k_1 k_2} \cdots d_{k_{r-1} j}$ and hence is a term of $d_{ij}{}^{(r)}$.

From these observations we see that each of the states of M in S_i has the same allowable input sequences and the same associated output sequences as does state i of N. Thus all of these states are equivalent (by Definition 1, Section VI) to the state i of N, $i=1, 2, \cdots, q$, and *vice versa*. Hence, by Definition 2 of Section VI, the machines M and N are equivalent. Since the machine N is uniquely defined, we may now summarize our observations thus:

Theorem 3: Given a machine M, there exists a corresponding equivalent machine N with minimum number of states which is unique to within a permutation of the states.

This theorem is the same as that of Mealy[10] which is an adaptation of a result of Moore.[11]

[8] Mealy, *op. cit.*, p. 1054.
[9] Moore, *op. cit.*, p. 143.
[10] Mealy, *op. cit.*, p. 1054.
[11] Moore, *op. cit.*, p. 142.

XI. An Example

To illustrate the ease of application of the preceding algorithm, even in the case of a complicated machine, we now apply it to the following connection matrix of a machine with 13 states, 6 inputs a, b, c, d, e, f and 4 outputs $\alpha, \beta, \gamma, \delta$:

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	(e, δ)	(b, γ)	0	0	0	0	0	$(a, \alpha) \vee (c, \beta)$	0	0	0	0
2	0	(a, β)	(e, β)	0	0	0	0	0	(c, β)	(d, α)	(b, γ)	0	0
3	(e, δ)	0	(b, γ)	0	0	0	0	0	(a, α)	0	(c, β)	0	0
4	(a, α)	0	0	(b, γ)	0	0	0	0	(e, δ)	0	(c, β)	0	0
5	0	0	(d, α)	(e, β)	(b, γ)	0	(a, β)	0	0	0	0	0	(c, β)
6	(e, α)	0	(f, γ)	0	0	(a, β)	0	0	0	(b, δ)	0	(d, α)	0
7	$(c, \beta) \vee (e, \delta)$	(b, γ)	0	0	0	0	0	0	0	(a, α)	0	0	0
8	(c, β)	(a, β)	(e, β)	0	0	0	(d, α)	0	0	0	(b, γ)	0	0
9	(a, α)	0	(b, γ)	0	0	0	0	(e, δ)	(c, β)	0	0	0	0
10	(e, δ)	0	0	0	0	0	0	(b, γ)	(c, β)	(a, α)	0	0	0
11	(c, β)	$(b, \gamma) \vee (a, \beta)$	0	0	0	0	(d, α)	0	0	0	0	(e, β)	0
12	0	(e, δ)	0	(b, γ)	0	0	0	0	(c, β)	0	0	(a, α)	0
13	0	0	(f, γ)	0	0	0	(b, δ)	0	(d, α)	0	0	(e, α)	(a, β)

Examining the rows of C, we first partition the states into the following maximal, disjunctive sets

$$S_1 = \{1, 3, 4, 7, 9, 10\ 12\}, \qquad S_2 = \{2, 5, 8, 11\}, \qquad S_3 = \{6, 13\},$$

such that each submatrix formed from the corresponding rows of each S_i is a 1 matrix. Next, partition C with respect to the S_i.

	1	3	4	7	9	10	12	2	5	8	11	6	13
1	0	(b, γ)	0	0	$(a, \alpha) \vee (c, \beta)$	0	0	(e, δ)	0	0	0	0	0
3	(e, δ)	(b, γ)	0	0	(a, α)	0	0	0	0	0	(c, β)	0	0
4	(a, α)	0	(b, γ)	0	(e, δ)	0	0	0	0	0	(c, β)	0	0
7	$(c, \beta) \vee (e, \delta)$	0	0	0	0	(a, α)	0	(b, γ)	0	0	0	0	0
9	(a, α)	(b, γ)	0	0	(c, β)	0	0	0	0	(e, δ)	0	0	0
10	(e, δ)	0	0	0	(c, β)	(a, α)	0	0	0	(b, γ)	0	0	0
12	0	0	(b, γ)	0	(c, β)	0	(a, α)	(e, δ)	0	0	0	0	0
2	0	(e, β)	0	0	(c, β)	(d, α)	0	(a, β)	0	0	(b, γ)	0	0
5	0	(d, α)	(e, β)	(a, β)	0	0	0	0	(b, γ)	0	0	0	(c, β)
8	(c, β)	(e, β)	0	(d, α)	0	0	0	(a, β)	0	0	(b, γ)	0	0
11	(c, β)	0	0	(d, α)	0	0	(e, β)	$(b, \gamma) \vee (a, \beta)$	0	0	0	0	0
6	(e, α)	(f, γ)	0	0	0	(b, δ)	(d, α)	0	0	0	0	(a, β)	0
13	0	(f, γ)	0	(b, δ)	(d, α)	0	(e, α)	0	0	0	0	0	(a, β)

Examining the rows of C once again, we form the following partitioning of the states into maximal, disjunct sets

$$S_{11} = \{1, 9, 12\}, \quad S_{12} = \{3, 4\}, \quad S_{13} = \{7, 10\}, \quad S_{21} = \{2, 8, 11\}, \quad S_{22} = \{5\}, \quad S_{31} = \{6, 13\},$$

again such that the submatrices determined by the S_{ij} are 1 matrices. Now, repartition the connection matrix with respect to the S_{ij}'s:

	1	9	12	3	4	7	10	2	8	11	5	6	13
1	0	$(a, \alpha) \vee (c, \beta)$	0	(b, γ)	0	0	0	(e, δ)	0	0	0	0	0
9	(a, α)	(c, β)	0	(b, γ)	0	0	0	0	(e, δ)	0	0	0	0
12	0	(c, β)	(a, α)	0	(b, γ)	0	0	(e, δ)	0	0	0	0	0
3	(e, δ)	(a, α)	0	(b, γ)	0	0	0	0	0	(c, β)	0	0	0
4	(a, α)	(e, δ)	0	0	(b, γ)	0	0	0	0	(c, β)	0	0	0
7	$(c, \beta) \vee (e, \delta)$	0	0	0	0	0	(a, α)	(b, γ)	0	0	0	0	0
10	(e, δ)	(c, β)	0	0	0	0	(a, α)	0	(b, γ)	0	0	0	0
2	0	(c, β)	0	(e, β)	0	0	(d, α)	(a, β)	0	(b, γ)	0	0	0
8	(c, β)	0	9	(e, β)	0	(d, α)	0	(a, β)	0	(b, γ)	0	0	0
11	(c, β)	0	(e, β)	0	0	(d, α)	0	$(b, \gamma) \vee (a, \beta)$	0	0	0	0	0
5	0	0	0	(d, α)	(e, β)	(a, β)	0	0	0	0	(b, γ)	0	(c, β)
6	(e, α)	0	(d, α)	(f, γ)	0	0	(b, δ)	0	0	0	0	(a, β)	0
13	0	(d, α)	(e, α)	(f, γ)	0	(b, δ)	0	0	0	0	0	0	(a, β)

Since the submatrices are not *all* 1 matrices we form a new partitioning of the states:

$$S_{111} = \{1, 9, 12\}, \quad S_{121} = \{3, 4\}, \quad S_{131} = \{7, 10\}, \quad S_{211} = \{2, 8\}, \quad S_{212} = \{11\}, \quad S_{221} = \{5\}, \quad S_{311} = \{6, 13\}.$$

Repartitioning the connection matrix with respect to the S_{ijk} gives:

	1	9	12	3	4	7	10	2	8	11	5	6	13
1	0	$(a, \alpha) \vee (c, \beta)$	0	(b, γ)	0	0	0	(e, δ)	0	0	0	0	0
9	(a, α)	(c, β)	0	(b, γ)	0	0	0	0	(e, δ)	0	0	0	0
12	0	(c, β)	(a)	0	(b, γ)	0	0	(e, δ)	0	0	0	0	0
3	(e, δ)	(a, α)	0	(b, γ)	0	0	0	0	0	(c, β)	0	0	0
4	(a, α)	(e, δ)	0	0	(b, γ)	0	0	0	0	(c, β)	0	0	0
7	$(c, \beta) \vee (e, \delta)$	0	0	0	0	0	(a, α)	(b, γ)	0	0	0	0	0
10	(e, δ)	(c, β)	0	0	0	0	(a, α)	0	(b, γ)	0	0	0	0
2	0	(c, β)	0	(e, β)	0	0	(d, α)	(a, β)	0	(b, γ)	0	0	0
8	(c, β)	0	0	(e, β)	0	(d, α)	0	(a, β)	0	(b, γ)	0	0	0
11	(c, β)	0	(e, β)	0	0	(d, α)	0	$(b, \gamma) \vee (a, \beta)$	0	0	0	0	0
5	0	0	0	(d, α)	(e, β)	(a, β)	0	0	0	0	(b, γ)	0	(c, β)
6	(e, α)	0	(d, α)	(f, γ)	0	0	(b, δ)	0	0	0	0	(a, β)	0
13	0	(d, α)	(e, α)	(f, γ)	0	(b, δ)	0	0	0	0	0	0	(a, β)

Since all submatrices are 1 matrices, the states corresponding to each S_{ijk} are equivalent states.

The reduced connection matrix is determined by replacing each submatrix by a single element which is the union of all elements in each submatrix. We have then the matrix

$$C' = \begin{array}{c} \\ 1 \\ 3 \\ 7 \\ 2 \\ 11 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 3 & 7 & 2 & 11 & 5 & 6 \end{array} \\ \begin{bmatrix} (a,\alpha) \vee (c,\beta) & (b,\gamma) & 0 & (e,\delta) & 0 & 0 & 0 \\ (a,\alpha) \vee (e,\delta) & (b,\gamma) & 0 & 0 & (c,\beta) & 0 & 0 \\ (c,\beta) \vee (e,\delta) & 0 & (a,\alpha) & (b,\gamma) & 0 & 0 & 0 \\ (c,\beta) & (e,\beta) & (d,\alpha) & (a,\beta) & (b,\gamma) & 0 & 0 \\ (c,\beta) \vee (e,\beta) & 0 & (d,\alpha) & (b,\gamma) \vee (a,\beta) & 0 & 0 & 0 \\ 0 & (d,\alpha) \vee (e,\beta) & (a,\beta) & 0 & 0 & (b,\gamma) & (c,\beta) \\ (d,\alpha) \vee (e,\alpha) & (f,\gamma) & (b,\delta) & 0 & 0 & 0 & (a,\beta) \end{bmatrix} \end{array}$$

which defines the simplest equivalent machine.

XII. Conclusion

In this paper we have developed and illustrated a systematic method for reducing a given state diagram of a sequential machine to its simplest equivalent form. The method is so organized that it could readily be programmed for a computer, which would reduce the labor significantly in problems of design. (The input-output pairs could be identified by the integers 1, 2, $\cdots$, mp for this purpose.)

In machines in which input restrictions exist one can at times combine nonequivalent states without altering the input-output behavior of the machine, provided appropriate input restrictions are observed. This problem, and the problem of synthesis, will be treated in subsequent papers.

Acknowledgment

The authors wish to thank Professor Sundaram Seshu of Syracuse University for his contribution to the development of their ideas. The authors are also grateful for the support of this research provided by the College of Engineering, University of Illinois.

Correspondence

The Logical Combination of Punched Paper Tapes*

The early design of a digital computer may call for its output section always to supply such "extras" as line numbers, plus and minus signs, tabs, and carriage return, in order to bring the numerical results of computation into a form that ultimately can be transcribed to typewritten copy. This strict editorial action may hamper the programmer in his effort to meet the requirements of an unknown [illegible].

Consider, for example, the problem of printing a typewritten number field one-hundred and twenty characters wide from computer-edited output consisting of ten useful characters per word. Algebraic signs, word endings, and tabs must be deleted to adjoin the proper sequence of characters without overflowing the carriage of the tape-controlled typewriter. Fig. 1 gives an idea of such copy as it ordinarily appears from the computer. Fig. 2 shows the contour map that was obtained from this copy by voiding the signs, eleventh digits, and between-word tabs during the operation of transcription. This reduction was accomplished easily by having the automatic typewriter (*e.g.*, tape-controlled Flexowriter) read an overlay tape along with the answer tape while typing the information.

This procedure may be explained as follows. The representation of each output symbol is by a six-hole binary code punched in a traverse row across the paper tape. A smaller feed hole is punched continuously near the center of the tape for the purpose of guiding the tape past the punched tape reader. In reading two paper tapes simultaneously, one above the other, normally open contacts for each bit are closed whenever a hole is sensed common to both tapes for that unit of the code. This is equivalent to finding the logical "and" of the two tapes. To have the automatic typewriter behave as though recognizing a delete, the corresponding position on the overlay tape is not punched. Consequently, the two tapes have

* Received by the PGEC, August 3, 1957.

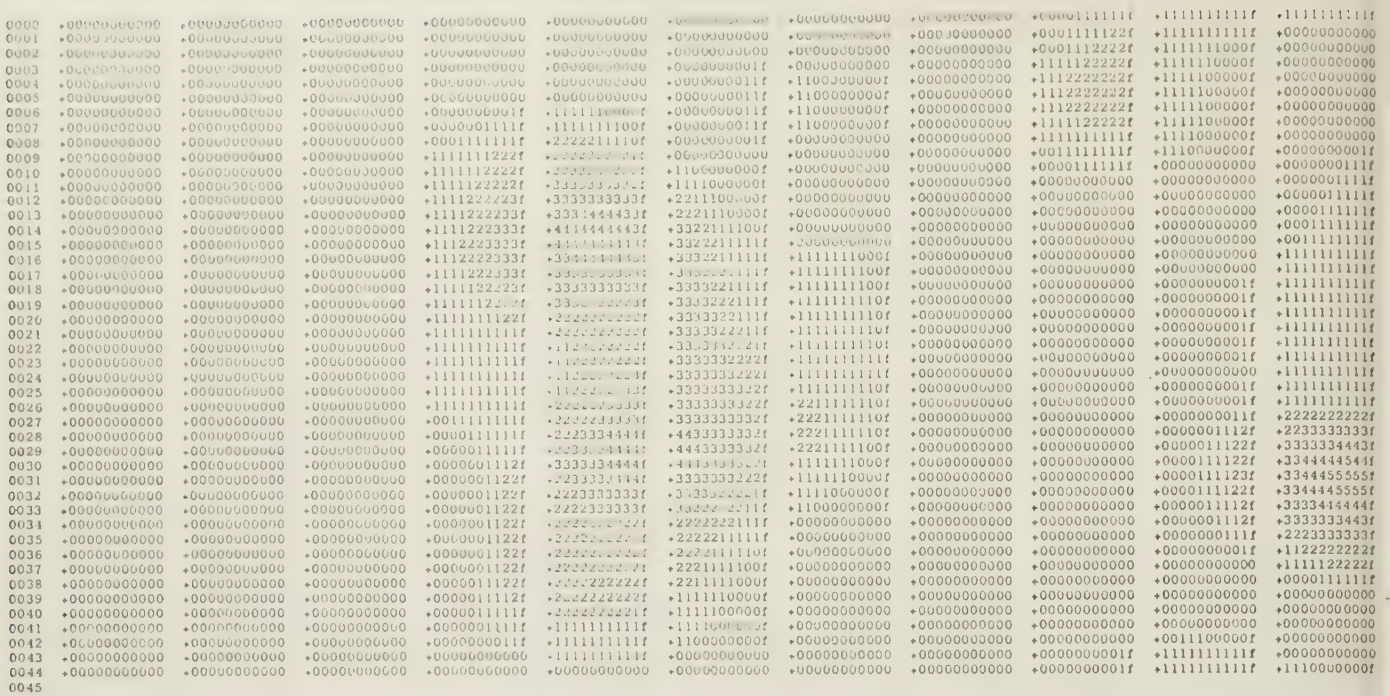


Fig. 1

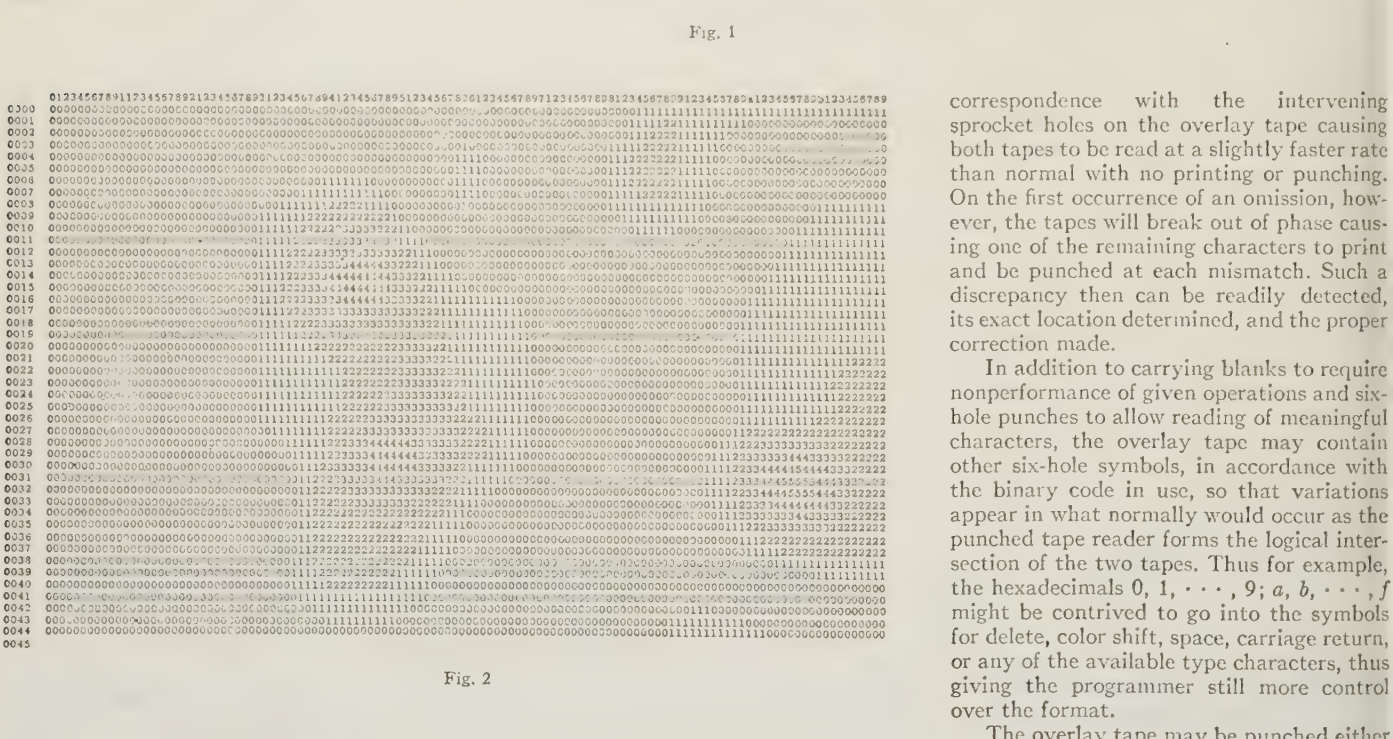


Fig. 2

just the sprocket hole in common, which indicates to the machine that no action whatsoever should be taken. To allow the meaningful characters to come through, all six holes are punched in the corresponding positions on the overlay tape.

The overlay principle can also be used in checking input tapes for the accidental omission of punched codes if some arbitrary code, such as carriage return, follows a predetermined pattern. The carriage returns, say, are replaced by six-hole punches on the overlay tape with the proper number of intervening sprocket holes to fill out the pattern. If the pattern is periodic, the overlay tape may be cemented to form a loop for convenience. Assuming the punch to be "on" and a sheet of paper wrapped completely around the platen for recording any error indication, the operation proceeds in the following manner. In the region where no mistake has yet occurred, the six-hole punches on the overlay tape will align on the carriage returns of the input tape with the resulting intersection causing the carriage to advance and a carriage return code to be punched. The remaining symbols of the input tape will be placed in one-to-one correspondence with the intervening sprocket holes on the overlay tape causing both tapes to be read at a slightly faster rate than normal with no printing or punching. On the first occurrence of an omission, however, the tapes will break out of phase causing one of the remaining characters to print and be punched at each mismatch. Such a discrepancy then can be readily detected, its exact location determined, and the proper correction made.

In addition to carrying blanks to require nonperformance of given operations and six-hole punches to allow reading of meaningful characters, the overlay tape may contain other six-hole symbols, in accordance with the binary code in use, so that variations appear in what normally would occur as the punched tape reader forms the logical intersection of the two tapes. Thus for example, the hexadecimals 0, 1, $\cdots$, 9; a, b, $\cdots$, f might be contrived to go into the symbols for delete, color shift, space, carriage return, or any of the available type characters, thus giving the programmer still more control over the format.

The overlay tape may be punched either by manual keyboard operation or by special perforating equipment. Somewhat similar effects may be obtained by placing the answer tape under the punch and the overlay tape on the reader. The resulting superposition is the logical "or" of the two tapes. It is also possible that the operations of tape union and intersection described could be applied in a mechanical way to derive truth tables for Boolean expressions in several variables.

Robert M. Mason
U. S. Naval Research Lab.
Washington, D. C.

Demonstration of Conditional Stability on an Analog Computer*

It is instructive to demonstrate experimentally a conditionally (Nyquist)-stable circuit. This is conveniently done on an analog computer. The setup turns out to be quite critical and some preliminary calculations must be carried out. This note presents the results.

Fig. 1 gives a Nyquist diagram of a conditionally stable system. If the critical point $(-1+0j)$ is at A or at C, the feedback system whose loop transfer function is $\overline{H}$ is stable; if it is at B, the system is unstable.

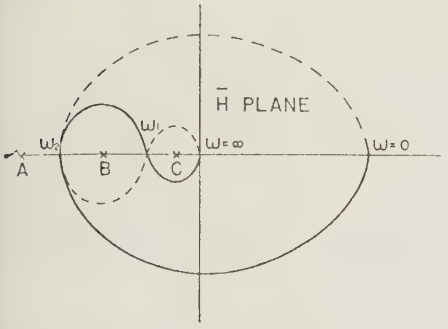


Fig. 1—Nyquist diagram of conditionally stable feedback system.

Starting with a loop gain which corresponds to B as critical point, and therefore to an unstable system, an *increase* in gain can shift the critical point to C and stabilize the system.

Perhaps the simplest transfer function exhibiting conditional stability is

$$\overline{H}(j\omega) = \frac{(1+j\alpha\omega)^2}{(1+jk\alpha\omega)^3}. \tag{1}$$

The locus of $\overline{H}(j\omega)$ cuts the real axis at

$$I_m\{\overline{H}(j\omega)\} = 0. \tag{2}$$

The corresponding angular frequencies ω_1, ω_2 are

$$\omega_{1,2} = \frac{k^2 - 6k + 3 \pm \sqrt{(k-1)^3(k-9)}}{2\alpha^2 k^2}. \tag{3}$$

Therefore, $k>9$ for $\overline{H}$ to correspond to Fig. 1. On the other hand, k must not be much larger than 9 if $\overline{H}(j\omega_2)/\overline{H}(j\omega_1) < \sim 10$. The frequency range for the experiment is determined by α,

$$\alpha \cong \frac{1.3}{\sqrt{k}\sqrt{\omega_1\omega_2}} \cong \frac{0.070}{\sqrt{f_1 f_2}}, \tag{4}$$

where the last expression follows for $k \cong 9$.

The following values correspond to Fig. 2–Fig. 4: $k=9.2$; $\sqrt{f_1 \cdot f_2} = 14$ cps. From (4) $\alpha = 5 \cdot 10^{-3}$ and from (3) $\omega_1 = 100$ cps, $\omega_1 = 38$ cps. Eq. (1) now gives $\overline{H}(j\omega_1) = -|\overline{H}(j\omega_1)| = -1.17\times10^{-2}$ and $\overline{H}(j\omega_2) = -2.66\times10^{-2}$, corresponding to loop gains of $A_1 = -1/\overline{H}(j\omega_1) = 85$ and $A_2 = -1/\overline{H}(j\omega_2) = 38$. These are the gains at the transitions from the unstable region to the stable region.

The complete setup is given in Fig. 4. $\tau = R_i C_i = 5\times10^{-3}$ sec; $k = R_f C_f / R_i C_i = 0.046$ $\alpha = 9.2$. The maximum loop gain is 153.

* Received by the PGEC, June 7, 1957.

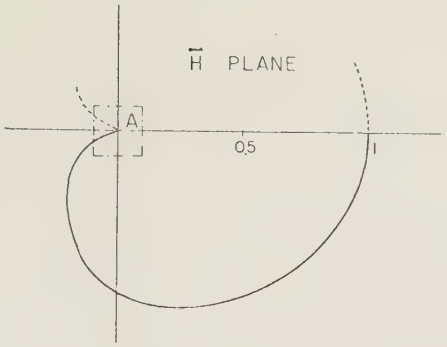


Fig. 2—Nyquist diagram of analyzed system, drawn to scale.

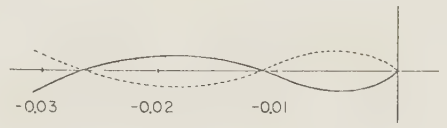


Fig. 3—Part A of Fig. 2, enlarged.

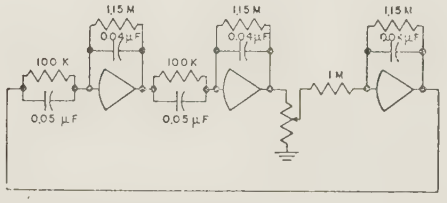


Fig. 4—Computer setup for system simulation.

The results are best observed in the phase plane representation on a cathode-ray oscilloscope. Horizontal and vertical deflection inputs are taken from the outputs of two of the computing amplifiers of the setup. A stable system is recognized by an inward going spiral, an unstable system by a growing spiral.

Amos Nathan
Yona Mahler
Faculty of Elec. Eng.
Technion, Israel Institute of Technology
Haifa, Israel

On the Use of Redundant Integrators in Analog Computers*

One type of instability in analog computer setups may be traced to the use of integrators whose total number exceeds the degree of the characteristic equation being simulated. This phenomenon has been reported before[1] but appears not to be well understood. Ideally, the use of redundant integrators should not affect the solution, but in practice it is found that extra integrators cause perturbations in the locations of the roots of the characteristic equation and may introduce extraneous roots with large positive real parts. These statements are demonstrated in the simple circuit of Fig. 1. Its circuit equations are

$$\begin{bmatrix} e(t) \\ 0 \end{bmatrix} = \begin{bmatrix} S+1 & -S \\ -S & S+1 \end{bmatrix} \times \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}.$$

A possible setup is shown in Fig. 2.

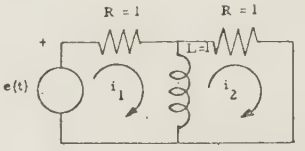


Fig. 1

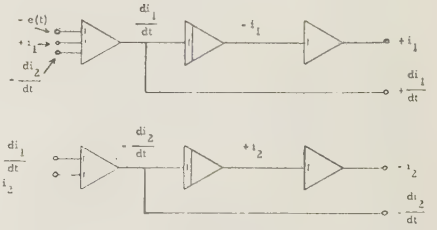


Fig. 2

Now let us suppose that the integrator in the upper chain has a multiplying factor of $1/k$ instead of 1. The characteristic equation of the computer setup is then

$$\begin{vmatrix} (1+kS) & -kS \\ -S & (1+S) \end{vmatrix} = 0$$

or

$$S(k+1) + 1 = 0.$$

If $k=1$, the computer setup yields the same characteristic equation as the original network, but if $k \neq 1$, the transient response will be of the form

$$i = A\epsilon^{(-t/k+1)} \text{ instead of } i = A\epsilon^{(-t/2)}.$$

This difference arises from the fact that the computer setup uses two integrators to do a job done in the network with only one. The total inductor current $(i_1 - i_2)$ equals the integral of one quantity, the inductor voltage. In the computer, however, i_1 is produced by one integrator and i_2 by another. If these are not perfectly matched, $(i_1 - i_2)$ will not be correct.

Mere dislocation of the roots of the characteristic equation, however troublesome it may be, is likely to occur even in the optimum computer setup due to component inaccuracies. These inaccuracies become even more troublesome, however, in setups which employ redundant integrators since there is a very real possibility that the degree of the characteristic equation may be increased with the result that not only are the desired roots moved but new ones are introduced. For example, suppose that the summing resistor to which $-di_2/dt$ is fed at the input of the upper chain is of such a value that this input is multiplied by a factor k instead of by unity.

* Received by the PGEC, October 11, 1957.

[1] L. G. Walters, "Hidden regenerative loops in electronic analog computers," IRE Trans., vol. EC-2, pp. 1–4; June, 1953.

The characteristic equation of the setup is then

$$\begin{vmatrix} (1+S) & -kS \\ -S & (1+S) \end{vmatrix} = 0$$

or

$$S^2(1-k) + 2S + 1 = 0.$$

The roots are

$$S = \frac{-1 \pm \sqrt{k}}{1-k}$$

for $k \neq 1$. If k is precisely one, then no trouble arises since the characteristic equation is of the first degree with its root where it belongs at $S = (-1/2)$. If k is slightly less than unity, the trouble is not severe since one of the roots is a large negative number, corresponding to a rapidly decaying exponential term which can be neglected, and the other root lies very close to $S = (-1/2)$. However, k is as likely to be slightly larger than one, and in this case the extraneous root is found to lie far out in the right-half plane. This root has a disastrous effect on the solution of the problem since it corresponds to a rapidly growing exponential term.

Such artifices as judiciously inserting additional damping to counteract the effects of spurious roots have been suggested, but these fail to recognize that the additional roots are present because the computer setup uses in its closed loop a number of energy-storage elements (the integrators) which exceed the degree of the characteristic equation of the problem. Extra integrators not included in the closed loop cause no trouble and may sometimes be required in cases where the minimal integrator setup to implement the equation yields only derivatives of some of the desired quantities.

NORMAN R. SCOTT
University of Michigan
Ann Arbor, Mich.

Contributors

D. D. Aufenkamp, for biography see page 205 of the September, 1957 issue of these TRANSACTIONS.

❖

Joseph O. Campeau was born in Cleveland, Ohio on March 10, 1930. He received the B.S. degree in electrical engineering from Case Institute of Technology, Cleveland, Ohio in 1952 and the M.S. degree from the University of California at Los Angeles in 1955. At the present time, he is studying for the Ph.D. at the University of California at Los Angeles.

From 1952 through 1956, he was engaged primarily in the design of digital computers at Hughes Aircraft Company, Culver City, Calif. Since 1956, he has been employed at Litton Industries, Beverly Hills, Calif. His work there is concerned with systems and digital computer design.

❖

C. K. Chow was born in Hong Kong, on December 28, 1928. He attended the Utopia University and National Tsing Hua University, China and was awarded the B.S.E.E. degree in 1949. He continued his education in the United States, and received the M.E.E. and Ph.D. degrees from Cornell University, Ithaca, N. Y., in 1950 and 1953, respectively.

From 1953 to 1955, he was an assistant professor in electrical engineering at the Pennsylvania State University, State College, Pa. Since 1955, he has been associated with the Burroughs Corporation Research Center, Paoli, Pa.

Dr. Chow is a member of Sigma Xi, Pi Mu Epsilon, and Phi Kappa Phi.

❖

D. W. Davies was born in Treorchy, Wales, on June 24, 1924. He attended the Imperial College of Science and Technology, London, Eng. and was graduated in 1943, with the degree of B.S. in physics. In 1946, he received the B.S. degree in mathematics from the same school. During World War II, he was engaged in theoretical work on the British atomic bomb project. In 1946, he joined the National Physical Laboratory. He took part in the design and construction of the ACE pilot model which began to work in May, 1950. At the present time, Mr. Davies is concerned with the design of the ACE computer.

❖

Robert J. Domenico was born in New York, N. Y., on August 1, 1926. He was graduated from Syracuse University with a B.S.E.E. degree in 1951.

He joined IBM in June, 1951, where he worked on the design of circuits for a gas tube counter, the design of a fast rise time pulse generator, and the design of a video amplifier for electrostatic memory. In 1954, he was assigned to the transistor circuit group to design logical switching circuits. In 1956, he was appointed to head a group in the Poughkeepsie Product Development Laboratory to study methods of mechanizing the design of transistor switching circuits.

❖

Paul C. Dow, Jr. was born in Melrose, Mass. on March 31, 1927. He received the B.S. degree from the United States Military Academy from which he graduated as a second lieutenant in the Air Force in 1949.

After completing flying training and serving two years as a flying instructor, he attended the guided missile course at the University of Michigan, receiving the M.S.E. degree in 1954. He served two years as a graduate assistant in aeronautical engineering at the Air Force Institute of Technology while completing his doctoral dissertation, and received the Ph.D. degree from the University of Michigan in 1957.

Captain Dow is presently assigned as a project officer at the Air Force Ballistic Missile Division of the Air Research and Development Command, Inglewood, Calif.

He is a member of Phi Kappa Phi, Sigma Xi, Institute of the Aeronautical Sciences, American Society for Engineering Education, and the National Society of Professional Engineers.

❖

Franz E. Hohn, for biography see page 206 of the September, 1957, issue of these TRANSACTIONS.

❖

Robert E. Horn (S'52–A'55) was born in Richmond Heights, Mo., on November 30, 1927. He received the B.S.E.E. degree from Washington University, St. Louis, Mo., in 1950. Awarded a Fortesque Fellowship for graduate study, he received the M.S.E.E. degree in 1951. Previously, he served two years as an electronic technician with the U. S. Navy.

From 1951 to 1955, he was an instructor in the electrical engineering department of Washington University. In addition to his teaching duties, he was responsible for installation and operation of the department's analog computing equipment. After obtaining the doctorate degree in 1955, he was employed by the Air Arm Division, Westinghouse Electric Corporation, Baltimore, Md., as a senior engineer engaged in airborne fire-control system studies. In his present position as a fellow engineer, he is responsible for the activities of an analytical group working on missile seeker and bomber defense problems.

Dr. Horn is a member of the AIEE, Association Internationale Pour le Calcul Analogique, Tau Beta Pi, and Sigma Xi.

❖

Verne G. Fauque was born in Kevin, Mont., on June 7, 1931. He received the B.A. degree in mathematics in 1953 and the M.A. degree in 1954 from Montana State University at Missoula.

In 1954, he joined the Westinghouse Electric Corporation's Graduate Student Training Program. Upon completing this program in 1955, he was assigned to the Air Arm Division, Baltimore, Md., where he is now engaged in fire-control system studies.

Mr. Fauque is a member of Pi Mu Epsilon.

PGEC News

Membership Distribution of PGEC by Sections

The membership of the PGEC increased 1226 from January 1, 1957, to August 15, 1957, when the total paid membership was 6551. Noticeable increases were seen in practically every section with outstanding gains in foreign memberships.

The distribution of paid members as of August 15, 1957, is shown as follows.

In IRE Sections

Akron	34
Alamogordo-Holloman	13
Albuquerque-Los Alamos	38
Atlanta	47
Baltimore	114
Bay of Quinte	3
Beaumont-Port Arthur	2
Binghamton	62
Boston	506
Buenos Aires	3
Buffalo-Niagara	31
Cedar Rapids	25
Central Florida	62
Central Pennsylvania	16
Chicago	189
China Lake	10
Cincinnati	41
Cleveland	42
Columbus	18
Connecticut Valley	121
Dallas	36
Dayton	50
Denver	29
Detroit	138
Elmira-Corning	4
El Paso	21
Emporium	4
Evans-Owensboro	6
Florida West Coast	3
Fort Huachuca	1
Fort Wayne	17
Fort Worth	31
Hamilton	5
Hawaii	2
Houston	46
Huntsville	34
Indianapolis	23
Israel	14
Ithaca	14
Kansas City	24
Little Rock	6
London, Ont.	7
Long Island	330
Los Angeles	891
Louisville	7
Lubbock	4
Miami	14
Milwaukee	54
Montreal	47
Newfoundland	1
New Orleans	15
New York	748
North Carolina-Virginia	37
Northern Alberta	4
Northern New Jersey	301
Northwest Florida	4
Oklahoma City	13
Omaha-Lincoln	5
Ottawa	18
Philadelphia	512
Phoenix	46
Pittsburgh	59
Portland	15
Princeton	32
Regina	0
Rochester	39
Rome-Utica	35
Sacramento	6
St. Louis	37
Salt Lake City	8
San Antonio	20
San Diego	59
San Francisco	341
Schenectady	31
Seattle	53
Shreveport	4
South Bend-Mishawaka	11
Southern Alberta	3
Syracuse	77
Tokyo	30
Toledo	8
Toronto	47
Tucson	12
Tulsa	19
Twin Cities	134
Vancouver	10
Washington	301
Wichita	9
Winnipeg	4

In Other Localities

Alaska	1
Argentina	3
Australia	5
Austria	2
Belgium	7
Bermuda	1
Brazil	4
Cuba	1
Czechoslovakia	2
Denmark	4
England	40
France	21
Germany	10
Greece	1
Hong Kong	1
India	4
Italy	17
Japan	30
Lebanon	1
Mexico	3
Netherlands	10
Norway	2
Scotland	1
Sweden	22
Switzerland	7
Turkey	2
Venezuela	3
USSR	1
Total Paid Members	6551

1958 Simulation Conference

The 1958 National Simulation Conference, sponsored jointly by the PGEC and the Dallas IRE Section, will be held in Dallas, Texas, on October 23–25, 1958. Full information regarding submission of papers and other details will be published in a future issue of these Transactions.

Summary of PGEC Administrative Committee Meeting

The PGEC Administrative Committee met during WESCON in San Francisco, on August 22, 1957. A quorum of 14 members and proxies out of a total of 17 members were present.

Secretary-Treasurer H. W. Nordyke reported membership and financial figures as of June 30, 1957:

Members	5998
Students	544
Receipts (January 1–June 30)	$16,776.44
Expenditures (January 1–June 30)	$ 9,339.46
Balance (June 30)	$26,356.99

Editor J. P. Nash reported that H. D. Huskey is resigning as editor of the Review of Current Literature section in the Transactions, effective January, 1958. L. F. Jones, Chairman of the Baltimore Chapter, and R. W. Melville agreed to cooperate in organizing a Bibliography Committee which would recommend action on the reviews as well as on a possible bibliography of computer literature.

On the motion of R. W. Melville, Student Relations Chairman, $100 was voted to his committee for working with high schools.

In conformance with IRE policy, a citizenship restriction which was inadvertently placed last year on the IRE Computer Fellowship will be removed in the future.

The AIEE has been added to the list of societies whose members may become PGEC affiliates.

The Administrative Committee decided that the PGEC Vice-Chairman should automatically be Chairman of the Awards Committee each year. This action assures that awards activities will continue to be in the hands of a member with necessary experience in PGEC affairs.

Other Committee appointments are:

R. W. Melville—Student Relations
L. C. Nofrey—Chairman, Membership
R. T. Blakely—Vice-Chairman, Membership
J. C. LaPointe—Constitution and By-Laws
N. M. Blachman—Publications
S. B. Disson—Sectional Activities
W. D. Winger—Technical Program Representative, 1958 IRE National Convention
J. P. Nash—Transactions Editor.

W. Buchholz
Chairman

Reviews of Current Literature

It is the intention of this section to review articles that have been published since January 1, 1953, and to publish eventually reviews of all books pertaining to the computer field. Authors can be of considerable assistance in this review process by sending two reprints of their articles to H. D. Huskey, Department of Electrical Engineering, University of California, Berkeley, California. The editors wish to express their gratitude to the reviewers who, through their efforts, make this section possible.—*H. D. Huskey*

GENERAL

57-159

The Computing Revolution—E. L. Harder. (*Elec. Eng.*, vol. 76, pp. 476–481, 586–590; June, July, 1957.) This two-part article outlines some of the significant developments in the computing field during 1956. Subjects discussed include computer developments, engineering applications, automatic programming, simulation, analog vs digital computers, computer uses in business, and control applications. The use of computers as an engineering and research tool is examined, and the growing uses in business and manufacturing are noted.

H. T. Larson

57-160

The Position of the University in the Field of High Speed Computations and Data Handling—A. S. Householder. (*Computers and Automation*, vol. 5, pp. 6–10; May, 1956.) In a rather rambling manner, the author describes the past and present activities of European and American universities, followed by a summary of his opinion of the proper future position of universities with respect to this subject. The main theme is that specialized courses in the programming of digital computers and numerical analysis are not the proper approach. He prefers the complete reorganization of the mathematics curriculum in a manner which integrates numerical methods with other techniques starting with the first-year calculus courses. It is the author's feeling that, if this is done, the student can receive a more sophisticated training in this technique than occurs in one semester courses which can only provide a collection of recipes. In this manner the student would be better prepared to obtain solutions to his mathematical problems in the field in which he is engaged.

Gordon E. Morrison

57-161

The Convention on Digital-Computer Techniques—(*J. IEE*, vol. 2, pp. 475–489; August, 1956.) These pages contain information on a Convention on Digital-Computer Techniques, April 9–14, 1956. Included are a three-page summary of the convention and reviews of the following papers: "Digital Computers and the Load-Flow Problem," by J. M. Bennett; "The Use of the Pilot Ace for Testing a New Design of Proton Synchroton," by G. G. Alway; "The Application of Digital Computers to Electric Traction Problems," by A. Gilmour; "The Use of Digital Computers in Obtaining Solutions to Electric-Circuit Problems Involving Switching Operations," by S. J. M. Denison and D. G. Taylor; "Sorting of Data on an Electronic Computer," by D. W. Davies; "Application of Digital Computers in the Exploration of Functional Relationships," by G. E. P. Box and G. A. Coutie; "An Electronic Calculator for Punched-Card Accountancy," by L. Knight; "The Programme-Controlled Computer," by E. J. Guttridge and R. P. B. Yandell; "The HEC Computer," by R. Bird; "The Manchester University Mark II Digital Computing Machine," by T. Kilburn, D. B. G. Edwards, and G. E. Thomas; "A Transistor Digital Computer," by E. H. Cooke-Yarborough, R. C. M. Barnes, J. Stephen, and G. A. Howells; "Transistor Arithmetic Circuits for an Interleaved-Digit Computer," by R. C. M. Barnes, G. A. Howells, and E. H. Cooke-Yarborough; "The Digital Computer as an Aid to the Electrical Design Engineer," by B. Birtwistle and B. M. Dent (See Rev. 57–218, this issue.)

D. E. Hart

681.142 57-162

Application of Computers in Automatic Systems—A. A. Fel'dbaum. (*Automatika i Telemekhanika*, vol. 17, pp. 1046–1056; November, 1956.) A survey. Twenty-six references including several to Russian literature.

Courtesy of Proc. IRE and *Wireless Engineer*

57-163

Analog Versus Digital Techniques for Engineering Design Problems—D. B. Breedon. (IRE TRANS., vol. PT-2, pp. 86–89; April, 1957.) Nearly every problem encountered in engineering at some time proceeds from the qualitative to the quantitative phase where the results of mathematical analysis must be applied in actual computation. Most often the computation is short enough that automatic means are not necessary. However, more and more problems are requiring powerful aids to calculation. This increase is due as much to expanded thinking encouraged by the mere availability of computers as to any actual backlog of work. Therefore it is to the engineer's advantage to know what computers can do for him, even though he may take his problem to someone else for final preparation and programming. The following text presents some examples in which automatic calculation is being used. The logic used in choosing the computing methods is shown based on the characteristics of problem and computer. As background for the examples the most important of these characteristics are presented briefly.

Author's Summary
Courtesy of PROC. IRE

57-164

A Shaft-to-Digital Encoder—B. M. Gordon, M. A. Meyer, and R. N. Nicola. (*Proc. Western Comp. Conf., February 11–12, 1954, Los Angeles, Calif.*; pp. 128–133; April, 1954.) A miniature commutator-type, analog-to-digital converter is described. Coding ambiguity is avoided by using two sets of commutators driven through an intermittent drive system and mechanically phased so that one commutator set is always stationary and ready for reading while the other commutator set is being positioned. Provision is made for connecting two input voltage sources to represent the zeros and ones of the parallel output. An auxiliary commutator is used to switch the input voltages to the stationary readout-commutator set. All commutators are cylindrical, one-quarter inch in diameter. Over-all size of encoder is $2\frac{1}{2}$ inches long and $1\frac{1}{2}$ inches in diameter.

R. A. Gaskill

57-165

Digital Compensation for Control and Simulation—Julius Tou. (PROC. IRE, vol. 45, pp. 1243–1248; September, 1957.) A very brief review of the Z-transform method is given. A mathematical treatment is given, illustrated by an example to show that 1) a digital-analog feedback system can be stabilized by digital techniques, 2) the system error can be made zero at the sampling instants by digital compensation, and 3) the programs required for 2) can be made realizable (*i.e.*, to require no foreknowledge) by a delaying program. The technique of synthesizing programs from Z transforms is not discussed nor are the physical requirements of the resulting programs (program steps, computing and sampling rates). The limitations of the technique do not seem to be clearly described.

A. D. Scarbrough

57-166

Temperature Dependence of Junction Transistor Parameters—Wolfgang W. Gartner. (PROC. IRE, vol. 45, pp. 662–680; May, 1957.) This paper presents calculated typical values of a wide assortment of transistor parameters as a function of temperature for four representative kinds of transistors

This page has been left blank in order that readers may mount all reviews on cards.
—*The Editor*

(Ge *p-n-p* alloy, Ge *n-p-n* grown, Ge *n-p-n* rate grown, and Si *n-n-n* grown). The curves presented should prove particularly valuable as indicators of expected trends to those engineers designing transistor circuits for military applications.

A. D. Scarbrough

681.142.002.2 57-167

Factory for Electronic Digital Computers—(*Engineer, London*, vol. 202, p. 593; October 26, 1956.) Design features of the "Pegasus" and "Mercury" computers are discussed.

Courtesy of Proc. IRE
and *Wireless Engineer*

ANALOG COMPONENT RESEARCH

57-168

An Electronic Analog Multiplier—D. C. Kalbfell. (IRE Trans., vol. EC-6, pp. 100–103; June, 1957.) This multiplier uses the variable pulse area principle, but employs phase sensitive circuitry to operate naturally in all four quadrants without bias voltages. The output is zero if either input is zero. The X and Y channels are separately linearized with independent feedback loops. The circuitry is simple and lends itself to either transistors or vacuum tubes.

Author's Summary
Courtesy of Proc. IRE

57-169

A Function Multiplier—J. D. N. Van Wyk. (*Nature*, vol 178, pp. 1247–1248; December 1, 1956.) An electronic multiplier is described which has 1 per cent nominal accuracy. A rectangular area is displayed on the face of a cathode-ray tube. The illumination in each of four quadrants of the rectangle is detected by phototubes. If the displacement of the rectangular area in the X and Y directions is proportional to two voltages, the output of the phototubes can be summed to yield the product of these voltages.

D. E. Hart

57-170

A New Diode Function Generator—T. Miura, H. Amemiya, and T. Numakura. (IRE Trans., vol. EC-6, pp. 95–100; June, 1957.) With the diode function generators that are currently in use, generation of functions is made by combining straight lines. The main drawback of these function generators is that the slope of the line segment cannot be changed independently. With the new function generator described in this paper, functions are generated by connecting independent line segments. Accordingly, the slope of each segment is given independently and also quantitatively. It is possible to approximate any desired function without recourse to an oscilloscope for inspection. These advantages are realized by using ganged potentiometers differentially. The operating principle and a practical generator experimentally built are described.

Author's Summary
Courtesy of Proc. IRE

ANALOG SYSTEMS RESEARCH

57-171

Dynamic Accuracy as a Design Criterion of Linear Electronic-Analog Differential Analyzers—A. Nathan. (IRE Trans., vol. EC-6, pp. 74–86; June, 1957.) A frequency error analysis of computing elements is presented which leads to a definition of their dynamic accuracy. The concept of a computing transfer function is introduced for this purpose, permitting the evaluation of an effective bandwidth, the latter being connected with the variance of the output for wide-band inputs. Limited bandwidth is considered as equivalent to finite resolution and thus to an additional effective error. Single frequency errors are dealt with separately and are shown to be of minor importance. Suitable optimalization of dynamic accuracy yields parameters of design and performance such as optimum computing time and required base amplifier gain. The theory is applied to integrators and adders with base amplifiers of direct and of capacitive coupling.

Author's Summary
Courtesy of Proc. IRE

ANALOG EQUIPMENT

57-172

QC Computers for Machine Control, Parts 1 and 2—G. H. Amber and P. S. Amber. (*Elec. Mfg.*, pp. 80–88, 78–85, 298; July, August, 1956.) Analog computers for computing $\bar{x}$ and σ for either continuous or discontinuous distribution are described. These computers can be used in automated systems to control quality automatically.

Courtesy of Data Processing Group
General Motors Research Staff

681.142 57-173

A New Analogue Computer using Matrix Iteration for Determining the Roots of Algebraic Equations—J. Miroux. (*Ann. Telécommun.*, vol. 11, pp. 226–232; November, 1956.) The mathematical principles underlying an experimental computer are detailed. Its application and possible development are discussed.

Courtesy of Proc. IRE
and *Wireless Engineer*

UTILIZATION OF ANALOG EQUIPMENT

57-174

Computers—The Key to Modern Manufacturing Scheduling—J. P. J. Gravel and T. F. Kavanagh. (IRE Trans., vol. PT-2, pp. 90–93; April, 1957.) Manufacturing operations with poor scheduling plans are headed for trouble. Load capacity analysis is a technique for measuring the feasibility and desirability of proposed scheduling plans. Simply stated mathematically, load capacity analysis consists of a series of multiplications and additions. However, the numerous computations in a typical problem usually take more time than can be allowed. The paper describes a special purpose analog computer specifically designed to solve this problem in a matter of minutes.

Author's Summary
Courtesy of Proc. IRE

57-175

Electronic-Analogue-Computer Study of Synchronous-Machine Transient Stability—A. S. Aldred and P. A. Doyle. (*Proc. IEE*, vol. 104, pt. A, pp. 152–160; April, 1957.) Authors' Summary: The paper describes a method of solution of synchronous-machine transient-stability problems by a more exact electronic analog representation of the system equations than has hitherto been employed. In the machine analysis emphasis is placed on the variation of field flux linkage during transient disturbances, and reasons are given why this is necessary. The equations of performance are derived in the most convenient form for analog computing. The equivalent analog interconnections are given both for time varying and for constant field flux linkage. The components required for these analogs are considered briefly. The section on computer control deals with the control of integrators with reference to the introduction of "initial conditions," "hold," and "compute" states. A delay unit is described which enables the switching out of a faulted transmission line to be simulated. The results of various problems solved by the computer are presented in graphical form as boundaries between stable and unstable states. It is shown how instability subsequent to the first rotor oscillation may occur, and an explanation is proposed to account for this phenomenon. Synchronous-machine and transmission-line analysis are excluded from the main text for clarity and are presented in detail in the Appendixes.

D. E. Hart

681.142 57-176

Approximate Solution of Differential Equations with Partial Derivatives using Electrical Analogues—E. S. Kozlov and N. S. Nikolaev. (*Automatika i Telemekhanika*, vol. 17, pp. 890–896; October, 1956.) Analogs for solving Laplace, Poisson, and Fourier-type equations are briefly discussed.

Courtesy of Proc. IRE
and *Wireless Engineer*

57-177

Computing Techniques for the Sampling Parametric Computer—C. J. Hirsch and F. C. Hallden. (IRE Trans., vol. EC-6, pp. 108–119; June, 1957.) This paper describes novel calculating techniques suitable for an electronic analog computer using exponential discharges to simulate the logarithmic scales of the slide rule. Among others, the device can perform the following operations: $y=xy$; $z=x/y$; $z=xy^a$; $\log_a x$; a^x; and evaluate such series as $y=Ax^a+Bx^b+Cx^c+\ldots$ where the exponents can be fractional and z, x, and y can be time variables. The problems can be solved explicitly or implicitly; thus in the above series y (or x) can be determined with equal ease if x (or y) are given. The appendix describes an actual computer which gave accuracies of 1 to 2 per cent of full range from 0 to 95 per cent of full range and 4 per cent from 95 to 100 per cent of full range without recalibration for different groups of problems. The accuracy can be increased by calibrating the device for a specific group of problems.

Author's Summary
Courtesy of Proc. IRE

This page has been left blank in order that readers may mount all reviews on cards.
—*The Editor*

57-178

Trigonometric Resolution in Analog Computers by Means of Multiplier Elements—R. M. Howe and E. G. Gilbert. (IRE Trans., vol. EC-6, pp. 86–92; June, 1957.) A method of generating sine and cosine functions in analog computers by means of multiplier elements and integrators is discussed. Static accuracy of the method is analyzed and found to be essentially equal to the accuracy of the multipliers employed. The system accepts $d\theta/dt$ as the input and generates output voltages of $\sin\theta$ and $\cos\theta$. Amplitude-stabilizing loops are employed to maintain $\sin^2\theta+\cos^2\theta=1$. Advantages of the method include representation of unlimited range in angles, dynamic capabilities far beyond that of the multipliers alone, and possibility of employing electronic multipliers. The method has been successfully used to compute Euler angles in analog computer solutions of the three-dimensional flight equations.

Author's Summary
Courtesy of Proc. IRE

DIGITAL COMPONENT RESEARCH

57-179

Basic Logic Circuits for Computer Applications—G. W. Booth and T. P. Bothwell. (*Electronics*, vol. 30, pp. 196–200; March, 1957.) Basic digital computer circuits designed around high-frequency transistors are described. The operating and design features of a flip-flop, gated pulse amplifier, dc amplifier, power pulse amplifier, and neon lamp indicator are discussed. The flip-flop and pulse amplifiers were designed to operate at a pulse repetition rate of 250 kc. The Eccles-Jordan circuit was chosen for the flip-flop because of its complimentary outputs. Saturated operation of the transistors is employed with collector current limited to 10 ma. Diode forward voltage drops and breakdown voltage of diodes and transistors dictated the choice of flip-flop output swing and gating levels of zero and minus 6 v. Further discussion deals with the type of logical structure for which the circuitry is best suited. Data regarding operating experience are presented and were gained through use of the circuits in connection with a general purpose computer which was specifically designed as a test facility.

Norman F. Loretz

57-180

Magnetic Computer has High Speed—T. H. Bonn. (*Electronics*, vol. 30, pp. 156–160; August, 1957.) The ferroactor, a magnetic amplifier used in the UNIVAC magnetic computer, is capable of operating at frequencies as high as 2.5 mc. The basic circuit is a series-type magnetic amplifier. Variations of this basic circuit produce complementing and noncomplementing amplifiers. A carrier cycle is divided into input and output periods. The noncomplementing amplifier produces signals during output periods only when signals are applied during corresponding input periods. The complementing amplifier normally produces output signals when no input signals are applied. An input signal causes a no output during the following output period. The no-output current of the amplifier is absorbed by a constant current sink, thus suppressing the no-output signal. Details of circuit operation and interconnection are presented. Factors affecting the choice of 4–79 permalloy from among other materials are discussed.

Norman F. Loretz

57-181

A 2.5 Megacycle Ferractor Accumulator—R. D. Torrey and T. H. Bonn. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 50–53; 1957.) Magnetic amplifiers having power gains up to four at an information frequency of 2.5 mc are described. Two types are shown, a straight amplifier used primarily to provide a pulse time delay and an inverting type. Each amplifier is a series pulse type with a cycle of operation divided into an input period and an output period. There is an inherent pulse time delay of 0.2 μsec through the amplifiers. Logical functions are performed by isolation diodes at the input of the amplifiers. By alternately defining a "one" as being the presence and then the absence of a pulse series chain of the inverting amplifiers will yield the logical configurations "and," "or," "and," "or," etc. A blocking pulse of half the clock pulse amplitude is used at the output. This blocking pulse disconnects the clock during the input phase and is arranged to have no effect during the output phase. The problems of clock generation and distribution are discussed and detailed information is shown for the clock transformer. The demonstration model chosen is an accumulator consisting of a unit adder and a four-bit circulating register. Provision is made to add unity when a push button is depressed; the accumulated count is displayed by means of incandescent lamps powered directly from the cores. Thirty-seven amplifiers are used in the model requiring a clock input power of approximately 15 w, or 400 mw per core.

Lad Andrews

57-182

A Saturable-Transformer Digital Amplifier with Diode Switching—E. W. Hogue. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 58–64; 1957.) This paper describes a digital amplifier employing a saturable transformer operating in the 100-kc to 300-kc range. The clock is a two-phase sine-wave voltage source of approximately 10-v amplitude. The over-all power consumed is 0.27 w per amplifying stage. The amplifier is suitable for use with two-level diode gating logic to provide "and" and "or" logical functions between amplifying stages. The "not" function is obtained by using two transformers to obtain a complementing action. A criterion for stability, which insures that signals are not attenuated in long chains of amplifiers, is established. Factors affecting the gain of the amplifier are explored and the effects of winding reactances illustrated. An amplifier using a permalloy core transformer can drive 6 to 8 other stages, whereas ferrite core transformers are capable of driving only two others. A six stage circulating register with broad operating tolerances is described. Also described is a proposed logical package containing an "and" and "or" input circuit, and both a direct and complemented output.

R. D. Torrey

57-183

Multihole Ferrite Core Configurations and Applications—H. W. Abbott and J. J. Suran. (Proc. IRE, vol. 45, pp. 1081–1093; August, 1957.) A method is described using 4-hole transfluxors and a special winding configuration to permit blocking and unblocking pulses to have virtually unlimited amplitude. By threading the ac drive winding through the transfluxor and a suitably designed single-hole core it is shown to be possible to use voltage source drivers. By using these techniques plus a winding technique to compensate for lack of squareness in the ferrite hysteresis loop, the authors obtained operation from −50c to +180c with an ac signal unblock-to-block ratio of at least 25 db. A number of winding configurations employing multihole cores to perform complex operations are described. For example, a 6-hole core (called a logicor) can be used to make a binary half-adder, a 4-terminal odd parity checker, and other complex devices. The logical applications do not provide the compensating features which permitted wide temperature range operation of the 4-hole transfluxors. The experimental work used an ac signal frequency of 50 kc and pulse rates up to 20 kc.

A. D. Scarbrough

57-184

Ferrite Apertured Plate for Random-Access Memory—J. A. Rajchman. (*Proc. Eastern Joint Comp. Conf., December 10–12, New York*, pp. 107–115; 1957.) For some time the idea of a continuous media for digital data storage has been considered. This is the first report of significant success in this endeavor. The development of small prototype ferrite plates with a 16 by 16 (256) aperture pattern is described and the behavior pertinent to memory application is thoroughly discussed. Of particular importance to memory technology, however, are the interesting circuit advances described for use in the memory array. A pair of plates is used for each digit plane, one storing the zeros and the other the ones. This provides noise cancellation, giving a positive signal for a one and a negative signal for a zero. It also provides a constant load on the drivers, since one core is always switched in each digit plane. A two-wire system is also described which requires a word selection matrix in addition to the memory stack. One wire is plated on the ferrite and the other wire is threaded straight through all of the planes at once. This results in only one aperture pair in a digit plane being driven during the read operation which greatly reduces noise problems. The cost advantage attendant on this method is significant. Other interesting techniques are reported in the description of the memory media and access methods.

Robert A. Tracy

57-185

A Compact Coincident-Current Memory—A. V. Pohm and S. M. Rubens. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 120–123; 1957.) This

This page has been left blank in order that readers may mount all reviews on cards.
—*The Editor*

paper describes the present status of the work on evaporated magnetic film storage elements. It is a summary in nature as the extent of the work is much too large to be covered completely in a paper of this length. Within this restriction it is an excellent paper as the work is described concisely and accurately. The techniques described in the paper represent a major advance in the magnetic core storage art and it is fortunate that so competent a paper should have been written so early in the development of these techniques. The only slight criticism that can be made is that the authors have tended to convey the impression that they are describing a working system rather than one under development. For those interested in magnetic core storage techniques this paper will be a required addition to their library.

Raymond Stuart-Williams

57-186

Megabit Memory—R. A. Tracy. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 104–106; 1957.) An interesting new method of making magnetic cores and memory planes is described in this short but well-illustrated article. Thin strips of 4–79 molybdenum permalloy ($\frac{1}{8}$ inch wide$\times\frac{1}{2}$ inch long by $\frac{1}{8}$ mil thick) are wound directly on the selecting and sense wires to form a core position. This wrapping technique provides a minimum inside diameter and thus keeps the driving currents small. The performance of sample wound cores is compared with an S-3 ferrite core in graphs of signal-to-noise ratio and temperature dependence and in photographs of core output waveforms. The signal outputs were 30–40-mv peak amplitude. The higher Curie temperature metal tape core was less temperature sensitive and gave a greater signal-to-noise ratio then the ferrite. The evolution of a completely wired 20×20 memory plane is described and illustrated with photographs. All wires (*X*, *Y*, *Z*, and SENSE) are first assembled in their proper relationship and then the cores are wrapped on at the intersections by a semiautomatic machine. Further details of this machine wrapping process and how uniformity is maintained are not given in this article. Several advantages over ferrites are claimed and discussed that include: 1) less temperature sensitivity, 2) negligible strain sensitivity, 3) major hysteresis loop operation, 4) very good uniformity of tape from the same melt. Large differences exist between melts, but 10–20 million cores can be made from one 50-pound melt, 5) useful operation over a wider current range, 6) a factor of 10-cost advantage over ferrite cores. The author claims that these improvements now make million-bit matrix memories quite feasible. An interesting question and answer discussion section follows the article.

W. W. Lawrence, Jr.

681.142:621.375.3 57-187

A Magnetic-Amplifier Switching Matrix—D. Katz. (*Commun. and Electronics*, no. 24, pp. 236–241; May, 1956.) The circuit described translates a four-digit binary code into an electronically displayed decimal output.

Courtesy of Proc. IRE and *Wireless Engineer*

57-188

A Large-Capacity Drum-File Memory System—H. F. Welsh and V. J. Porter. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 136–138; 1957.) This paper describes the 2,500,000-character flying head drum memory designed for Univac-Larc, which provides a maximum access time of 2.6 seconds. The presentation is essentially a product description, outlining the basic design philosophy in terms of overall system requirements and the design steps leading to the present configuration, which includes an air floating head which is positional by a linear indexing mechanism over 100-information bands. The authors reveal a clear understanding of the formidable problem involved in providing a high-speed electromechanical random access memory and indicate significant progress towards a solution, which at present seems a better commercial answer than the use of multiple magnetic disks or hydraulically lubricated drums. The claim that close mechanical precision is eliminated by the air floating design seems somewhat misleading in the case of this particular drum since the complexity of the head positioning mechanism more than compensates. Also, in adopting such a head design, one trades temperature differential and control problems for new problems of extreme cleanliness and surface smoothness. In view of the intimate relationship among drum, head, and reading system the paper might well have discussed the associated recording system more extensively, particularly since excellent density and clock rates were achieved (400 bits /inch).

D. N. MacDonald

681.142:621.395.625.3 57-189

High-Density [magnetic-] Tape Recording for Digital Computers—(*Elec. Mfg.*, vol. 56, pp. 153, 290; November, 1955.) Digital pulse densities of up to 700/inch can be recorded by the method described which was developed by the National Bureau of Standards for use with the SEAC computer.

Courtesy of Proc. IRE and *Wireless Engineer*

621.318.57:621.387 57-190

Dekatron Drive Circuit and Application—M. Graham, W. A. Higinbotham, and S. Rankowitz. (*Rev. Sci. Instr.*, vol. 27, pp. 1059–1061; December, 1956.) "A reliable, one-tube drive circuit for decade glow-transfer counter tubes is described. Application of the circuit is illustrated in a ten-channel glow-tube register with automatic electric-typewriter readout."

Courtesy of Proc. IRE and *Wireless Engineer*

681.142 57-191

An Accumulator Unit for a Dekatron Calculator—R. Townsend and K. Camm. (*Electronic Eng., London*, vol. 29, pp. 58–61; February, 1957.) Decimal numbers may be added or subtracted, the sum being returned to the main store multiplied or divided by ten or unity. The calculator has a punched-card input.

Courtesy of Proc. IRE and *Wireless Engineer*

621.385.832.032 57-192

The Technology of Electrostatic-Storage Cathode-Ray Tubes—P. Choffart. (*Onde élect.*, vol. 36, pp. 815–821; October, 1956.) An outline of some of the manufacturing problems.

Courtesy of Proc. IRE and *Wireless Engineer*

57-193

Polarization Reversal and Switching in Guanidinium Aluminum Sulfate Hexahydrate Single Crystals—H. H. Wieder. (Proc. IRE, vol. 45, pp. 1094–1099; August, 1957.) The above ferroelectric material, GASH, is compared to the more familiar barium titanite. For both crystals no threshold field and hence no true coercivity was found experimentally. The upper practical limit of driving frequency appears to be 25 to 50 kc for both crystals. GASH has lower hysteresis losses, very much lower dielectric constant, and lower electromechanical activity than barium titanite. The author feels that GASH has many properties of interest but that a great deal of additional work is required to establish its proper place as a reliable solid-state electronic component.

A. D. Scarbrough

57-194

Fast Switching by Use of Avalanche Phenomena in Junction Diodes—B. Salzberg and E. W. Sard. (Proc. IRE, vol. 45, pp. 1149–1150; August, 1957.) In this letter, the authors discuss the fast switching speeds in both "forward" and reverse directions which can be obtained by operating around the avalanche breakdown voltage. A number of oscillograms are presented which show extremely fast forward switching (transient undetectable with a Tektronix 545).

A. D. Scarbrough

681.142:621.314.7 57-195

The Junction Transistor as a Computing Element—E. Wolfendale, L. P. Morgan, and W. L. Stephenson. (*Electronic Eng.*, vol. 29, pp. 2–7, 83–87, 136–139; January, March, 1957.) The small-signal and transient characteristics of the transistor and their application to the design of basic circuits are described and examples of computer elements using transistors are given.

Courtesy of Proc. IRE and *Wireless Engineer*

57-196

High-Speed Transistor Computer Circuit Design—R. A. Henle. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 64–66; 1957.) This paper describes a novel switching-circuit philosophy particularly suited for the class of drift transistors with which appreciable improvement in switching speed may be obtained by using high currents and collector voltages. It is shown that high-speed operation demands small signal-voltage swings. These requirements are met by controlled-current switching, wherein a fixed current source supplies two or more transistor emitters in parallel. This entire current flows through that transistor with the most forward base

This page has been left blank in order that readers may mount all reviews on cards.
—*The Editor*

bias. Because emitter current is controlled, the collector current is limited, so that collector-voltage swings can be maintained within any desired range by suitable circuit impedances. This technique avoids low collector voltages and hole-storage delay at the expense of high transistor dissipation. This price may be great, for it is probable that the ultimate transistor reliability needed for large-scale computers will be achieved only with very small dissipation. Furthermore, this circuitry may require complementary *p-n-p* and *n-p-n* transistors without which some basic circuits, such as flip-flops, may be relatively complex. ("Millimicrosecond Transistor Current Switching Circuits"—Hannon S. Yourke, presented at the 1957 Transistor and Solid State Circuits Conference, Philadelphia, Pa., February, 1957.) Nevertheless, until devices yielding high speed in saturating circuits become available, controlled-current switching should fill many needs in the high-speed computer frontier.

J. B. Angell

57-197

High-Temperature Silicon-Transistor Computer Circuits—J. B. Angell. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 54–57; 1957.) The switching characteristics of the Philco T-1159 silicon surface alloy transistor are described as functions of temperature. It is shown that direct coupled transistor circuits can be made to operate satisfactorily from −50°C to +140°C with these transistors. Switching times of 0.1 to 0.2 μsec at room temperature and of about 0.5 μsec at +140°C are obtained in such circuits. The statements concerning the temperature stability of direct coupled circuits as compared to other types of circuitry are debatable, especially when the distribution of base drive currents due to different input impedances are considered.

W. B. Cagle

57-198

High Temperature Printed Circuitry—Quarterly Report No. 2, Third Series—J. R. Bowman, C. H. T. Wilkins, *et al.* (*Quart. Rep. Computer Components Fellowship Mellon Inst.*, 36+ix pp.; January 1, 1957–March 31, 1957.) Measurements of resistance, capacitance, and dissipation factor as a function of temperature over the range 28° to 260°C have been taken of some resistor and capacitor films. The film components tested were prepared by vacuum evaporation techniques and by the screen processing of enamel materials. Cycling the films in this temperature range showed that the components stabilized during the first cycle. Details are presented for the design of a continuous kiln for the firing of enamel components constructed by this laboratory. Of thirteen different solders tested for forming high temperature connections, only three were found to retain suitable mechanical and electrical properties after aging in an oven at 250°C for five weeks. Consideration is given to the development of pressure type connections for high temperature application.

C. H. T. Wilkins

57-199

High Temperature Printed Circuitry—Quarterly Report No. 3, Third Series—G. H. Young, C. H. T. Wilkins, *et al.* (*Quart. Rep. Computer Components Fellowship Mellon Inst.*, 21+vi pp.; April 1, 1957–June 30, 1957.) Enamel resistance films of bismuth fluxed and of glass fluxed gold have been found to have positive temperature coefficients between 25° and 500°C. The addition of TiO_2 to glass enamels has been demonstrated to increase the capacitance of printed capacitors by increasing the dielectric constant. A new extruded alumina sheet is being tested as a substrate material for use with both the screen printed enamels and the vacuum evaporated circuits. A preliminary study of the subzero properties of modular printed circuits has been undertaken. The method used and results obtained with vacuum evaporated gold-palladium resistance films in the range from −196°C to room temperature are presented. In the high temperature testing program, problems encountered in devising suitable soldering joints are discussed and present soldering procedure described. Difficulties encountered in using the large testing oven have led to the development of simplified heating units capable of testing one sample at a time up to temperatures as high as 850°C.

C. H. T. Wilkins

DIGITAL SYSTEMS RESEARCH

57-200

Arithmetical Analysis of Digital Computing Nets—Richard C. Jeffrey. (*J. Assoc. Comp. Mach.*, vol. 3, pp. 360–375; October, 1956.) The author indicates several possible numerical interpretations of a sequence of binary digits, where the sequence may be temporal or spatial. They differ according to the assumed location of the radix point and the representation of negative numbers. He then shows how the Boolean algebraic description of a computing net can be translated into an arithmetic description of its operation in terms of whichever interpretation is of interest.

A. S. Householder
Courtesy of *Mathematical Reviews*

57-201

An Algorithm for Determining Minimal Representations of a Logic Function—B. Harris. (IRE Trans., vol. EC-6, pp. 103–108; June, 1957.) For each logic function, or Boolean algebraic expression, there corresponds an appropriate computer circuit. However, the minimization of the appearances of the Boolean variables does not necessarily lead to the most economical circuit. A general approach to the problem therefore requires the development of techniques for the simple and rapid generation of a variety of near-minimal forms. This paper describes such a method for constructing the minimal representations of a logic function given as a truth table or in one of its canonical forms. The minimal representations achieved are either sums of products, or products of sums, such that no term contains superfluous variables and such that no term is superfluous. The utility of the method lies in the conciseness of notation, which permits the handling of a large number of variables and simplifies the process for machine computation.

Author's Summary
Courtesy of Proc. IRE

57-202

Minimization of the Partially-Developed Transfer Tree—M. P. Marcus. (IRE Trans., vol. EC-6, pp. 92–95; June, 1957.) A transfer tree is a particular type of multiterminal network having a single input which may be connected to any one of a number of outputs. An *n*-relay transfer tree is partially developed if it has less than the 2^n possible output terminals. Rearrangement of a partially developed tree can lead to a reduction in the total number of transfers required. This paper presents a method of obtaining a required partially developed transfer tree with the minimum number of transfers.

Author's Summary
Courtesy of Proc. IRE

57-203

A Time-Sequential Tabular Analysis of Flip-Flop Logical Operation—G. W. Arant. (IRE Trans., vol. EC-6, pp. 72–74; June, 1957.) In examining flip-flop response the principal concern of the logical designer is to find what input signals must be applied to the flip-flop in order to produce the output conditions that are desired. Equation methods of analysis and a time-sequential tabular method of analysis are described, and some advantages of the tabular method are pointed out.

Author's Summary
Courtesy of Proc. IRE

57-204

Synchronization of a Magnetic Computer—J. Kielsohn and G. Smoliar. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 90–93; 1957.) The digital signals which circulate in a high-speed synchronous computer are of short duration, have precisely determined shape and magnitude, and occur at precise times. Signals from computer input equipment are usually of much greater duration with undetermined shape and have random occurrence times as far as the computer is concerned. Therefore it is always necessary to have a device which will inject into the computer a sequence of properly shaped and timed signals which bears a one-to-one informational correspondence to the sequence of signals from the input equipment. Such a device is called a synchronizer. The authors describe a synchronizer employing diode gating and high-speed magnetic digital amplifiers of the series, or Ramey type, developed by Remington-Rand. The Cambridge computer, for which this synchronizer was designed, makes use of the same kind of magnetic amplifier. A brief description of the amplifier is given. Logically, the synchronizer is of familiar design. It consists of two dynamic flip-flops in cascade. The first flip-flop, when triggered by the unsynchronized signal, sends a series of pulses to the and-gate input of the second flip-flop. Coincident with the next computer timing pulse introduced at the other input to the and-gate, one of the pulses emitted by the first flip-flop attempts to trigger the

This page has been left blank in order that readers may mount all reviews on cards.
—*The Editor*

second. A full timing pulse period (two word lengths) is available for the second flip-flop to respond, or to fail to respond, before its output is sampled at the time of arrival of the next ensuing timing pulse at the output and-gate. Because the response time of the flip-flop is considerably less than the timing period, a fully shaped and synchronized output signal is guaranteed. A description of the operation of the magnetic shift register stage of the computer is also given in some detail.

E. W. Hogue

57-205

Designing for Reliability—Norman H. Taylor. (PROC. IRE, vol. 45, pp. 811–822; June, 1957.) A circuit design philosophy developed at Lincoln Laboratories is presented which tends to produce reliable circuit designs. The concept of marginal checking is discussed and illustrated by a great number of plots for a high-speed vacuum tube flip-flop. A number of useful rules of thumb for component derating is presented and the reliability of various components is discussed. Unfortunately, nothing helpful is said about transistors.

A. D. Scarbrough

57-206

Topology of Switching Elements vs Reliability—J. P. Lipp. (IRE TRANS., PGRQC-10, pp. 21–33; June, 1957.) The topology of switching elements is explored from a probability or reliability standpoint. Arrays are indicated which offer a greater degree of reliability than any one of their elements, hence disproving the belief that circuit complexity necessarily lessens reliability. A further result is production of a simple mathematical technique adaptable to the solution of complex circuits in terms of reliability. Such "circuits" may actually consist of redundant communication paths as a more general interpretation of switching elements.

Author's Summary
Courtesy of PROC. IRE

57-207

Improvement of Binary Transmission by Null-Zone Reception—F. J. Bloom, S. S. L. Chang, B. Harris, A. Hauptschein, and K. C. Morgan. (PROC. IRE, vol. 45, pp. 963–975; July, 1957.) This paper discusses the improvement in transmission of binary information in the presence of noise that can be effected by decoding into 3 or 4 states instead of the usual two. In the systems discussed the receiver would indicate state 1, state 2, or undecided (for single null reception); or state 1, state 2, undecided but probably state 1, or undecided but probably state 2 (for double null-zone reception). The use of null-zone reception permits a single parity check which normally gives single error detection to have substantially single error correcting properties since if the single error is registered as a null, the parity check can be used to correct the error. It is shown that single null reception greatly improves the effectiveness of the single parity check. Double null reception provides only a small additional improvement in per-unit equivocation but permits much less critical adjustment of the null level.

A. D. Scarbrough

DIGITAL EQUIPMENT

57-208

Die Mathematischen Grundlagen für die Organisation der Elektronischen Rechenmaschine der Eidgenössischen Technischen Hochschule—John Robert Stock. (*Mitt. Inst. Angew. Math. Zürich*, no. 6, 73 pp.; 1956.) This gives an account of the logical design of the ERMETH. This is a one-address serial floating decimal machine with a magnetic drum memory of 10,000 words of 16 digits. A floating decimal number is of the form $a \cdot 10^b$, where $-200 \leqq b \leqq 199$ and $|a| < 10$; alternatively a word can represent a fixed decimal number with 14 decimal digits. There is another decimal digit in the word, which serves two purposes as a parity check (mod 3) and as an aid (Q sign) for a programming system devised by Rutishauser [same *Mitt.* no. 3 (1952); MR 15, 64]. An instruction consists of 7 d.d.: four for the address, two for the operation, and a third indicating one of nine index registers. The machine arithmetically is roughly a 10-m one and has an average access time of about 5 m. The normal input is by means of punched cards. The first introductory chapter includes a list of the instructions. The second describes the actual form of the operations. The third chapter discusses the principles of realization of the machine. The fourth chapter is concerned with the logical design of the various components. There are appendixes which give block diagrams of the arithmetic unit and the control. There are a few simple examples of programming for the machine which illustrate the use of index register and the method of reading in a program; there is also a subroutine for square root and a program for the solution of a system of equations by an elimination process, which uses the index register and the Q sign.

J. Todd
Courtesy of *Mathematical Reviews*

57-209

The Transac S-1000 Computer—J. L. Maddox, J. B. O'Toole, and S. Y. Wong. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 13–16, 1957.) This paper presents a description of a scientific computer designed at the Philco Corporation. The machine uses the well-known direct-coupled surface-barrier transistor computer circuits reported on at the IRE Convention in March, 1955, by R. H. Beter, W. E. Bradley, R. B. Brown, and M. Rubinoff. Because of the packaging with printed circuits and the small volume associated with the transistor circuits the machine occupies only the usable space in a standard office desk. The memory, obtained from the ERA Division of Remington Rand UNIVAC, has 4096 words of 36 bits each contained in magnetic cores. The machine requires 1.2 kw of power. Much of the paper is concerned with the logical system of the machine and a list of instructions which has some similarity to the instruction list for a UNIVAC 1103A. The word length of 36 bits when used as an instruction contains six bits for the command, three bits for each of two address modifiers, and two 12-bit addresses.

Arithmetic is performed in the 1's complement parallel binary system by a 72-stage subtractor working with a 72-bit accumulator. A repeat instruction is available. The construction of the machine was 80 per cent finished when the paper was written. The machine may be considered to be a relatively straightforward application of the surface-barrier transistor circuits.

R. E. Meagher

57-210

TX-O, A Transistor Computer with a 256 by 256 Memory—J. L. Mitchell and K. H. Olsen. (*Proc. Eastern Joint Conf., December 10–12, 1956, New York*, pp. 93–101; 1957.) This two-part paper describes the TX-O magnetic core memory and a 5-megapulse transistor logic circuitry. The TX-O memory provides storage for 65,536 19-bit words. It is a parallel, conventional coincident-current magnetic-core unit having a 7.0-μsec read-write cycle. Random access to the memory is provided by two 256-position tape-wound, magnetic core switches. A block diagram of the memory system and a timing diagram are given. The memory and switch cores, the memory array windings and organization, and the design and operation of the access switches are fully described. The circuits of the memory system, which use 425 dual triodes and 625 transistors, are given with schematics. The system margins are shown. The TX-O utilizes transistor logic in two basic circuit configurations; the saturated inverter and the saturated emitter follower. Reverse base bias and speed-up condensers are used. Logic is performed by series and parallel combinations of these configurations. A quite complicated flip-flop circuit is used in an effort to have one standard flip-flop. Circuit schematics, waveforms, margins, and packaging are shown. While the description of the memory is quite complete, the description of the rest of the computer is restricted to the circuitry. Since neither the logical design nor the system characteristics are discussed the title of this paper is misleading.

J. A. Githens

681.142 57-211

The "Bizmac" Digital Data Processing System—J. C. Hammerton. (*Electronic Eng.*, vol. 29, pp. 174–180; April, 1957.) As used at the central agency for American military supply depots.

Courtesy of PROC IRE
and *Wireless Engineer*

57-212

Design Objectives for the IBM Stretch Computer—S. W. Dunwell. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 20–22; 1957.) This paper is a progress report on a very impressive computer system in its research phase. STRETCH is a multiple arithmetic unit computer system in the very large, ten-megapulse class. It is intended for both scientific and business applications. It consists of three separate stored program computer sections especially tailored to the requirements of handling 1) input-output, 2) data format editing, and 3) computation. All communicate with one multisection memory and all operate simultaneously to execute about one million instructions per second. Computer 1 will service individual input-output devices of a very

This page has been left blank in order
that readers may mount all reviews on cards.
—*The Editor*

wide class. Computer 2 is to be serial and handles variable word length entries in either binary or alpha-numeric notation. Basic speeds of this section are to be about 2–3 μsec for addition and 5–15 μsec for multiplication. Computer 3 is to be parallel and will perform floating point addition in 0.6 μsec and multiplication in 1.2 μsec. Housekeeping operations for this computer section proceed concurrently with computation. The instruction structure will allow for one million words of random access memory. External memory can be expanded to one hundred million words. The ten-megapulse solid-state logical elements and ferrite memories are described in other papers in the referenced proceedings. The fast memory will respond in 0.2 μsec. The project is reported to be substantially on schedule. The first system will go to Los Alamos.

William F. Gunning

57-213

A Transistorized Transcribing Card Punch—C. T. Cole, Jr., K. L. Chien, and C. H. Propster, Jr. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 80–83; 1957.) This paper describes a card punch used for the conversion of large volumes of data stored on magnetic tape in the Bizmac code into punched cards. The unit will transcribe information at a rate of 150 cards per minute and provides accuracy-checking features. Three cards' worth of information are in process at any one time—one set of information being read into storage, the second set being punched on the card, and the third set being read back from a punched card and compared with the storage. The punching and card handling techniques appear to be conventional in nature. The electronics of the unit are completely transistorized, with two-input resistor logic being used as the standard logical element. Combinations of these elements, which are basically a two input "or" gate, are utilized to generate an "and" gate (by inverting the inputs to the standard "or" gate), a "not" gate, an inverter, a power amplifier, and a flip-flop.

M. J. Mendelson

57-214

Automatic Input for Business Data-Processing Systems—K. R. Eldredge, F. J. Kamphoefner, and P. H. Wendt. (*Proc. Eastern Joint Comp. Conf., December 10–12, 1956, New York*, pp. 69–72; 1957.) The paper presents a semitechnical description of a machine which reads printed matter (presently, only numerical digits plus a few symbols) printed with magnetic ink, developed by the Stanford Research Institute. Described are three interrelated problems connected with the design of the equipment: 1) electrical and magnetic techniques for reliable reading of characters; 2) development of suitable magnetic ink for printing; 3) development of an electromechanical machine for reliably handling documents. The reading circuits employ a magnetic head with a gap width approximating the width of characters, a tapped electrical delay line allowing simultaneous examination of the electrical signals produced by the head over the period of time during which the character passes the reading head, and a correlation analysis network for recognizing the particular distribution of signals in the delay line. The printing of the information on the document is merely a good quality printing process employing a specific type of ink made up of a magnetic pigment. The document handling techniques employ a pneumatic system and are capable of handling documents which have been mutilated to quite an extent. The equipment has been employed principally in reading identification numbers from bank checks. Figures on rejection and error rate when reading these documents are presented. Following the interesting but brief presentation, a list of questions and answers serve to give further insight into this outstanding development.

L. S. Michels

57-215

A Tank Farm Data Reduction System—D. J. Gimpel. (IRE Trans., vol. PT-2, pp. 94–100; April, 1957.) A tank farm data reduction system has been developed for the new Tidewater Oil Company installation in Delaware by the Armour Research Foundation and Panellit, Inc. The function of the unit is to secure the temperature corrected volume of fluid in each of the approximately 100 tanks in the field. The inputs to the system are the fluid height and average tank temperature. Fluid volume is tabulated digitally as a function of the height on magnetic tape. The system automatically searches the tape for the indicated volume and multiplies the number by the temperature correction factor. This paper describes the operation of the multiplier, the tape search elements, and the sensing instruments employed in the field. The factors governing the selection of the specific elements in the storage and computing system are also discussed.

Author's Summary
Courtesy of Proc. IRE

681.142 57-216

A High-Speed Data Processing System—M. L. Klein, R. B. Rush, and H. C. Morgan. (*Electronic Eng.*, vol. 29, pp. 158–163; April, 1957.) Data from 20–100 independent channels, at a full-scale level of 100 mv, are taken at a rate of 100 kc and recorded digitally on a magnetic tape together with the source information and time.

Courtesy of Proc. IRE
and *Wireless Engineer*

UTILIZATION OF DIGITAL EQUIPMENT

57-217

Chemical Computing—(*Chem. Eng. News*, vol. 34, pp. 3542–3545; July 23, 1956.) This article gives a brief survey of current computer applications in engineering, data reduction, research, and personnel problems. It is of interest primarily to those unfamiliar with the computer field.

Courtesy of Data Processing Group
General Motors Research Staff

57-218

Digital Computer as an Aid to the Electrical Design Engineer—B. Birtwistle and Beryl M. Dent. (*Engineer*, vol. 201, pp. 440–442; May 4, 1956.) Two applications are given of the digital computer: 1) impulse voltage distribution in transformer winding, and 2) starting torque calculations for salient pole synchronous motors.

Courtesy of Data Processing Group
General Motors Research Staff

681.142:621.316.11 57-219

Automatic Network Analysis with a Digital Computation System—S. Y. Wong and M. Kochen. (*Commun. and Electronics*, no. 23, pp. 172–175; May, 1956. Discussion, p. 176.) A general outline of the procedure and a method of solution are given.

Courtesy of Proc. IRE
and *Wireless Engineer*

57-220

Fitting the Digital Computer into Process Control—Montgomery Phister, Jr., and Eugene M. Grabbe. (*Control Eng.*, vol. 4, pp. 129–136; June, 1957.) This article describes some of the techniques and problems involved in applying the digital computer to process control. The authors point out that in addition to knowledge of computer techniques a rather broad knowledge of the process itself is required. The authors describe a five step procedure of the systems development: 1) theoretical analysis of process, 2) correlation of variables, 3) invention, 4) specification of system components, 5) system operation. They give a design case history of a chemical process including a reactor, a heat exchanger, a catalyst separator, and a fractionating tower, and describe in some detail how the operational system can be synthesized. Also included is the development of the appropriate variable relationships from empirical data, together with diagrams of optimal operating loci. They describe how the computer program might be designed to accomplish the necessary control and how the digital control would be connected with the system.

Neal J. Dean

57-221

Tape Identification and Rerun Procedures for Tape Data Processing Systems—Leonard Eallson. (*Computers and Automation*, vol. 5, pp. 12–13; April, 1956.) This is a tutorial article which briefly describes widely used techniques for detection of certain tape handling errors and the recovery of the required information where reruns are necessary. The errors are detected by use of an information block on the tape containing identifying information, block counts, and other control information. The information for reruns is provided by a memory dump at the end of each tape output. This article is of no value to anyone familiar with routine data-processing procedures.

Gordon E. Morrison

57-222

The Digital Approximation of Contours—Robert M. Mason. (*J. Assoc. Comp. Mach.*, vol. 3, pp. 355–359; October, 1956.) The problem is to program a computer to trace a contour $f(x, y)=k$ on a grid. Suppose, for definiteness, $f<k$ inside the contour line. The sign of $f-k$ is examined at a sequence of points differing by Δx or by Δy. The sequence is rectilinear until $f-k$ changes sign, after which the direction rotates by $\pi/2$ if $f-k$ has become positive, $-\pi/2$ if $f-k$ has

This page has been left blank in order that readers may mount all reviews on cards.
—*The Editor*

become negative. For getting onto the contour initially the machine seeks out a path of steepest ascent or descent.

A. S. Householder
Courtesy of *Mathematical Reviews*

57-223

Some Experiences on the Manchester Computer with the Simplex Method—D. G. Prinz. (*Conference on Linear Programming*, Ferranti Ltd., London, pp. 73–89; May, 1954. Discussion, pp. 90–91.) This paper presents an account of the coding of the simplex method on the Manchester computer by the author. It is free from fussy details pertinent to a particular machine, but it does not shirk explanation of the numerical difficulties encountered by round of errors and how they were overcome. Mathematically, the most interesting point is the attempt to show some justification for the fact that the simplex method seems to take a number of iterations much lower than the conceivable (though never properly estimated) upper bound. The author arrives at the average result $\log_n C_m$ where m is the number of rows, n the number of columns of the tableau. The reasoning is heuristic and not particularly convincing, but it is, so far as the reviewer knows, the first brave effort to estimate the number of iterations.

A. J. Hoffman
Courtesy of *Mathematical Reviews*

681.142:51 57-224

Introduction to a Theory of Ensembles Based on the Prime Numbers Assimilable by Electronic Computers—S. Sabliet. (*C. R. Acad. Sci., Paris*, vol. 244, pp. 35–38; January 2, 1957.) Computers of "ordinator" type can be made to provide algebraic solutions of equations by coding the functions to be operated on as prime numbers.

Courtesy of Proc. IRE
and *Wireless Engineer*

BOOK REVIEWS

57-225

Elektronische Rechenmaschinen und Informationsverarbeitung—Edited by A. Walther. (Verlag Friedr. Vieweg & Sohn, Braunschweig, Germany, 229 pp.+viii, illus.; 1956.) Like many other proceedings of computer conferences, this volume covers a wide range of subjects, from the design to the application of digital computers. This one, however, has a decidedly cosmopolitan flavor. The authors come from eleven countries and their papers are printed in three different languages. The title *Electronic Digital Computing and Information Processing*, was that of a conference held in Darmstadt, Germany, in October, 1955. The organizer of the conference, Prof. A. Walther of Darmstadt, is also the editor of this book containing the sixty-four papers delivered at the conference. He has ably performed a most difficult job. The authors represent most of the major computing centers in Western Europe, as well as the United States and three eastern countries (the USSR, Czechoslovakia, and Eastern Germany). Prof. Walther has collected their papers into a readable and interesting volume. Forty-five of the papers are in German, seventeen in English, and two in French. English-speaking readers will find that the linguistic hurdles have been lowered by the inclusion of English abstracts for papers not in English. A curious sidelight is the fact that the two Soviet papers appear in German while the three Czech papers are printed in English. The book records an important conference and is thus of historical value. Many of the papers describe machines and techniques employed at various European computing centers. Others present research work of more general interest. The volume contains some unusual items, such as a demonstration that all arithmetic and logical computer operations can be built up from just one essential instruction containing only an address. The technical material will be largely familiar to those who have followed developments in the U. S. What is, perhaps, most interesting to such readers is the account of the many computer projects which are carried on in Europe, and the resourcefulness displayed in the face of limited budgets.

Werner Buchholz
Courtesy of Proc. IRE

57-226

Élektronnye Vychislitel'nye Mashiny i Obrabotka Informatsii (Electronic Computing Machines and Information Processes)—S. A. Lebedev. (*Akad. Nauk S.S.S.R., Moscow*, 47 pp.; 1956.) This pamphlet is one of the series of educational booklets issued by the Academy for popular consumption. Its author is the Academician in charge of BESM (this name is made up of the initials of the four Russian words standing for High-speed Electronic Computing Machine) and principal compiler of the Academy's Index of Mathematical Tables, Moscow, 1956 (Rev. 49, pp. 104–106, this issue). The booklet is couched in delightfully lucid language and is profusely illustrated with diagrams, photographs, and sample programs. The modest price, equivalent to about 15 cents, makes it available to anying having an interest in the subject, and its high quality guarantees complete satisfaction to each purchaser. The introduction points out the benefits of high-speed computation and emphasizes, in particular, the achievements made possible through the use of BESM not only in numerical computation but also in the initial—and quite spectacular—attempts to translate scientific English into Russian. The first chapter is devoted to a careful explanation of how electronic machines function. In the second, we find an excellent discussion of the binary vs the decimal systems with a complete explanation of the engineering principles which are used to represent the former system. This is followed by some very timely observations on the pros and cons of floating radix point representation of numbers. The last chapter deals with a detailed description of BESM. Since the features of this machine are already familiar to the readers of this magazine, there is no need to repeat the facts here.

The High-Speed Electronic Calculating Machine of the Academy of Sciences of the U.S.S.R.—S. A. Lebedev, *J. Assoc. Comp. Mach.*, vol. 3, pp. 129–133; July, 1956. See Rev. 57-24; March, 1957.) The pamphlet is enriched by an excellent appendix, which can easily serve as a textbook for tyrosin coding. It outlines the basic principles of program preparation, and illustrates them by actual examples. It touches upon the use of subroutines and the choice of suitable checks to insure the accuracy of the computations. An altogether charming booklet for 65 kopacks!

Ida Rhodes
Courtesy of *Mathematical Tables and Other Aids to Computation*

57-227

Digital Computer Programming—D. D. McCracken. (John Wiley & Sons, Inc., New York, N. Y., 218 pp.+30 appendices+5 index+vii; 1957.) Primarily an introductory text on digital computer programming, this book is also a survey of programming techniques. It is the first general treatment of programming without excessive concern for a particular aspects of the subject or for a particular computer. A successful exposition of computer programming must necessarily refer to a specific machine. The author has chosen to use a hypothetical computer, TYDAC. Like the alternative solution of using an existing computer, the decision to use TYDAC is not ideal but it is, however, skillfully implemented. The book remains a book on programming and not on programming TYDAC. The reader with no computer at his disposal will find TYDAC easy to understand and to program, although an element of reality is absent because TYDAC, unlike most real computers, has no peculiarities or irregularities. The reader with a computer will find that the illustrative programs do not depend on the detailed characteristics of TYDAC, that the exercises are readily adaptable to other machines, and that chapters not applicable to his machine may be omitted without loss of continuity. The author, fortunately, considers programming as the entire problem preparation effort and not merely as the detailed writing of machine instructions. The various aspects of programming are, with one exception, well covered. Such topics as subroutines and indexing are well developed. A brief introduction to automatic coding is included. The illustrations and exercises are very effective. The organization of the book suffers in the separation of decimal point location methods from floating decimal methods, but is otherwise successful. The text is written simply and is quite readable, despite occasionally faulty grammar or imprecise language. Although a good chapter on program checkout is included, the important companion subject of checks and reruns is not considered. Fuller discussion of the advantages and disadvantages of some of the various programming techniques presented would also represent an improvement. The book is well conceived, however, and generally well executed. It is a good introduction to an increasingly popular subject.

Peter Calingaert

57-228

Transistor Engineering Reference Handbook—H. E. Marrows. (John F. Rider, Inc., New York, N. Y., 288 pp.+2 indexes+viii, illus.; 1956.) The author states in the introduction that, while many publications deal-

This page has been left blank in order that readers may mount all reviews on cards.
—*The Editor*

ing with transistors are available, "the need to assemble and coordinate information on all the commercial aspects of the industry under one cover had led to the writing of this book." The volume contains five sections. Section I is a "General Survey of Transistors." Following a chronology of important transistor developments, a short description of transistor materials, structures, and fabrication techniques, and a list of pertinent references is given. Characteristics and circuit properties of junction transistors are discussed next. A classified bibliography on transistor applications concludes the section. The material is complemented by forty figures and tables related mainly to transistor circuit design. Section II occupies most of the volume and presents detailed data sheets (including characteristics and performance ratings) on some two hundred types of commercial transistors which were available at the time when the book was written. Section III contains reference data on circuit components used with transistors, such as audio transformers of various categories, IF and pulse transformers, capacitors, batteries, thermistors. A list of commercially available transistor test sets is also given. Section IV lists the specifications and, in many cases, shows the circuit diagrams of some one hundred products utilizing transistors. Products described are various amplifiers, pulse generators and other oscillators, power converters, meters, flip-flop, communication equipment, and radio receivers. The volume is completed by a Manufacturers' Directory (Section V). This reviewer is unable to display a great deal of enthusiasm for Section I. The material is too condensed to serve either as an understandable introduction to transistor physics and applications for the beginner or as a usable reference for engineers acquainted with the art. As the section stands, not even the little space occupied by it is used efficiently. For example, the author describes such experimental devices as filamentary transistors, field-effect transistors, and fieldistors but but makes no mention of drift transistors and unijunction transistors. The model circuits presented are not selected most judiciously to be representative of the contemporary art. There is some duplication: for example, instead of presenting a larger variety of flip-flops with a description of their respective merits, three essentially identical configurations are shown on pages 1-25; similarly, the configurations of Figs. 1-19 and 1-20 and those of Figs. 1-20A and 1-20B are identical. The author did a very conscientious and useful job in compiling the material of Sections II, III, and IV. These sections will be of considerable use to transistor circuit designers. It certainly is preferable to have information on transistors and associated components available in a single volume rather than to have to search for the material in countless specification sheets assembled in a multiplicity of folders. It should, of course, be remembered that, due to the steady release of new and improved transistors and components, the value of the book is necessarily ephemeral. However, for the present, it will be found useful by many engineers.

A. P. Stern
Courtesy of PROC. IRE

An Introduction to Cybernetics—W. R. Ashby. (John Wiley & Sons, Inc., New York, N. Y., 287 pp.+7 index+ix, illus.; 1956.) Dr. Ashby, who is known to many engineers as the author of *Design for a Brain* and the father of the "homeostat," has written what he considers an easy *introduction* to cybernetics and not "merely a chat about cybernetics." The preface informs us that the book addresses itself to the "many workers in the biological sciences" who "are interested in cybernetics and would like to apply its methods and techniques to their own specialty." It is Ashby's contention that the basic ideas of cybernetics can be treated without reference to electronics and that they are fundamentally simple. Ashby expresses, furthermore, the conviction that although advanced techniques may be needed for advanced applications, a great deal can be done in the biological sciences by the use of quite simple techniques, provided they are used with a clear and deep understanding of the principles involved. He expresses the belief that by refining common-place and well-understood concepts step by step, the biological worker who has no knowledge of mathematics beyond elementary algebra can be introduced to such cybernetics topics as feedback, stability, regulation, ultrastability, information, coding, noise, and so forth. Ashby divides his book into three major parts: Part I deals with the principles of mechanisms (including those that apply to determinate machines and very large systems). Part II, entitled, "Variety," deals with what is meant by information; it is designed to enable the reader "to proceed without difficulty to the study of Shannon's own work." Part III deals with regulation and control and provides "an explanation of the outstanding powers of regulation possessed by the brain, as well as the principles by which designers may build machines of like power." The book is well written and equally well produced (with the exception of a rather curious index which includes entries such as borrowed knowledge, chair chameleon, coffee, fly-paper, flying saucer ghosts, Hitler, etc.). And yet, this reviewer cannot rid himself of a certain uneasiness in recommending it for the purpose for which Ashby wrote it, to wit: for those who want to achieve by self-study an actual mastery of the subject. Many admittedly easy exercises are scarcely designed to relieve this reluctance. When, in the chapter entitled, "The Black Box," exercise 6/16-2 asks the question, "To what degree is the Rock of Gibraltar a model of the brain?," it is with a sense of keen disappointment that one reads the answer, "It persists, so does the brain; they are isomorphic at the lowest level." Such exercises hardly do justice to the often insightful exposition that Ashby provides elsewhere. The book's strongest point is, perhaps, that it brings the mathematically unsophisticated reader into some elementary contact with a broad spectrum of mathematical raw materials for conceptual models. In this respect Ashby's book fills a void in the cybernetic literature in spite of its exceptionally scant bibliography. Ashby's views of and aspirations for cybernetics as a mathematical discipline are revealed when he writes, "the truths of cybernetics are not conditional on their being derived from some other branch of science." In Ashby's view, cybernetics, whose subject matter is the domain of all possible machines, can, like mathematical physics be indifferent to the criticism that it gives prominence to the study of nonexistent systems. Here Ashby becomes a victim of his own semantics. The position of esteem that mathematical physics occupies today must in large part be attributed to the fact that throughout the past three centuries scientists have learned to abstract from their experience aspects that are both mathematically manipulable and yet relevant to predicting and controlling an ever-widening range of physical phenomena. Workers in the "soft" sciences (it might be more appropriate to call them the not-so-vertebrate sciences: their problems are tough rather than soft, but the fields lack a strong theoretical backbone) were deeply stirred by the cybernetic wave of the future. Here was a suggestion that they use new mathematical models and techniques that promised to be more applicable to problems of information, organization, perception, and perhaps even learning than the calculus has been. Here was a bridge to an era of computer technology with its potentialities of handling large quantities of data and of trying out complex conceptual models in reasonable time spans. Here was, finally, new hope for a rapprochement between scientists in these so-called soft areas with their more secure colleagues under the aegis of a possible unity of the communications sciences. Many of these hopes have been dashed, and much good will has been dissipated by those who honestly proclaimed that the problems of the biological and social sciences were on their way toward "solution" each time they had to their own satisfaction been able to transform "organized complexity" by the use of a more or less common-sense verbal device. This reviewer is convinced that we must not relax our efforts but that we must on the contrary dig in for a much longer pull. There are some tender shoots in areas in which models of modest scope and measurable phenomena are actually on speaking terms. We must intensify our explorations of mathematical models (such as in the theory of automata) and continue to search for significant experiments that will take advantage of our newly acquired armamentarium. There are passages in Ashby's book that are not in marked disagreement with the above views. And yet, the tone of the whole book does not seem to convey the proper perspective of the road ahead. There are scientists whose current work is more or less directly inspired by the writings of Wiener, Shannon, etc., Should their work be labeled as cybernetics and be introduced accordingly? This is certainly a matter of choice. Ashby has a perfectly good right to ignore these rather specific experiments and to substitute his own vision of the spectrum of problems that cybernetics will ultimately be capable of dealing with. But by not giving us at least one example of a truly successful application of a cybernetic model to a really tough biological problem, Ashby's book claims too much and proves too little.

W. A. Rosenblith
Courtesy of PROC. IRE

This page has been left blank in order that readers may mount all reviews on cards.

—*The Editor*

Index to

IRE TRANSACTIONS

ON

ELECTRONIC COMPUTERS

Volume EC-6, 1957

IRE Transactions on Electronic Computers

Index to Volume EC-6, 1957

Contents

Volume EC-6, Number 1, March, 1957

Volume EC-6, Number 2, June, 1957

Volume EC-6, Number 3, September, 1957

Volume EC-6, Number 4, December, 1957

Index to Authors

Numbers refer to index numbers in Contents listing

Index to Technical Subjects

Index to Nontechnical Subjects